An Evidential Deep Learning Classifier with an Integrated Capability for Uncertainty Quantification

by

Noofa Hammad

B.Sc., Effat University, 2020

A Thesis Submitted in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF APPLIED SCIENCE

in the Department of Electrical and Computer Engineering

University of Victoria

An Evidential Deep Learning Classifier with an Integrated Capability for Uncertainty Quantification

By

Noofa Hammad

B.Sc., Effat University, 2020

Supervisory Committee

Dr. Homayoun Najjaran, Supervisor
(Department of Mechanical Engineering)

Dr. David Capson, Supervisory Committee Member
(Department of Electrical and Computer Engineering)

ABSTRACT

While deep neural networks (DNNs) have demonstrated great proficiency in diverse tasks spanning various domains, the reliability of their predictions remains a subject of ongoing research. In the context of classification problems, there is a common misconception regarding probabilities generated by DNNs, falsely equating them with the confidence of the models in their assigned classes. Incorporating the softmax layer at the end of the network compels models to convert activations to probabilistic values between 0 and 1, irrespective of the underlying activation values. When activations are insufficient for accurate decision-making, raising uncertainty about the correct classification, a model should quantify its uncertainty about the true classification of the input data rather than making uncertain decisions. In this light, this study proposes a distance-based evidential deep learning (d-EDL) classifier with an additive capability for uncertainty quantification (UQ). The d-EDL classifier comprises two key components: the first utilizes convolutional neural network (CNN) layers for feature extraction, while the second incorporates designed layers for decision-making. In the second component, the first layer calculates basic probability assignments (BPAs) from the extracted feature vectors using a distance metric, measuring proximity between an input pattern and selected data representatives. A clustering algorithm is employed to form representatives for each data label; the closeness to a label representative reflects the potential belonging of the input to that label. The second and third layers employ combination rules to merge BPAs, leveraging probability theory and Dempster-Shafer (D-S) theory. The output of the d-

EDL network is a probability distribution extended to include uncertainty as a class. An end-to-end training method is provided to train the proposed classifier, enabling joint learning and updating all network parameters. Five variants of the d-EDL classifier, each with a different number of data representatives, are trained on an image dataset, and their uncertainty quantification ability is assessed. The assessment involves evaluating the models in three scenarios, each with a common misclassification leading factor: noise, image rotation, and out-of-distribution (OOD) data. The results demonstrate the excellent capability of d-EDLs, especially those with 20 and 40 data representatives, to effectively quantify uncertainty rather than misclassification when faced with unfamiliar data.

Table of Contents

List of Tables

List of Figures

Acronyms

AI	Artificial intelligence
BBB	Bayes by Backpropagation
BbH	Bayes by Hypernet
Bel	Belief function
BL	Bayesian learning
BNN	Bayesian neural network
BPA	Basic probability assignment
CNN	Convolutional neural network
CV	Computer vision
d-EDL	Distance-based evidential deep learning
DL	Deep learning
DNN	Deep neural network
DP-Fast MH	Differential Privacy-Fast Metropolis-Hastings
DR	Dropout Regulation
D-S	Dempster-Shafer
ECDF	Empirical cumulative distribution function
EDL	Evidential Deep Learning

EL	Evidential learning
EMNIST	Extended MNIST
FE	Feature extractor
GGN	Gauss-Newton
HMC	Hamiltonian Monte Carlo
KMNIST	Kuzishiji-MNIST
k-NN	k-nearest neighbors
L1	First layer
L2	Second layer
L3	Third layer
LA	Laplace approximation
LBD	Learnable Bernoulli Dropout
LLA	Linearized Laplace approximation
MAP	Maximum a posteriori
MC	Monte Carlo
MCMC	Markov Chain Monte Carlo
MDS	Multidimensional Scaling
m-EDL	Modified Evidential Deep Learning
MH	Metropolis-Hastings
ML	Machine learning
MNF	Multiplicative Normalizing Flows
MVG	Matrix Gaussian Posteriors

NN	Neural network
NBL	Non-Bayesian learning
OOD	Out-of-distribution
PBP	Probabilistic Backpropagation
Pl	Plausibility function
ReLU	Rectified Linear Unit
SG-MCMC	Stochastic Gradient Markov Chain Monte Carlo
SMH	Scalable Metropolis-Hastings
SSIM	Structural Similarity Index
Tanh	Hyperbolic Tangent
UQ	Uncertainty quantification
VI	Variational Inference
WCSS	Within-cluster sum of squares

Notations

S	Sample space
$\{\cdot\}$	Set
$\subseteq$	Subset of
E	Event
$P(\cdot)$	Probability
A, B, C	Events, hypotheses, or categories
$\cdot^c$	Complement
$\cup$	Union
$\cap$	Intersection
$\emptyset$	Phi or empty set
$P(\cdot \mid \cdot)$	Conditional probability
Ω	Frame of discernment, set of classes, or uncertainty class
ω_j	Data class
M	Total number of classes
P_i	Probability
2^{Ω}	Power set of Ω

$m(\cdot)$	Mass function
$[\cdot]$	Range
$Bel(\cdot)$	Belief function
$Pl(\cdot)$	Plausibility function
$BetP(\cdot)$	Pignistic probability
$\lvert\cdot\rvert$	Cardinality
$f(\cdot)$	Function
x_i	Input data or data point from the training set
$\boldsymbol{z}_i$	Feature vector
$(\cdot)$	Vector
T	Transpose
x	Data point from the test set
$\mathcal{X}$	Training set
$\in$	Belongs to
d_i	Distance
$\boldsymbol{p}_i$	Prototype (data representative) vector
Z	Set of feature vectors
Y	Training label set
n	Total number of data representatives
k_i	Number of class representatives or clusters
$\boldsymbol{c}_i$	Cluster centroid
$WCSS$	Within-cluster sum of squares

$\|\cdot\|$	Euclidean distance
$\boldsymbol{z}$	Feature vector for x
s_i	Monotonically decreasing function
exp (·)	Exponential function
α_i	Constant controlling the weight of exp (·)
γ_i	Constant controlling the spread of exp (·)
∞	Infinity
$\lim_{d\to\infty}$	Limit as d goes to infinity
$\boldsymbol{m}$	Mass function vector
E_i	Evidence source
$q_j(\cdot)$	First rule of combination
$q_j\_norm(\cdot)$	Normalized result for the first rule of combination
$\mu_j(\cdot)$	Second rule of combination
$\boldsymbol{\mu}$	Probability vector
$\boldsymbol{t}$	Ground truth label vector
t_j	True value for ω_j
v	Fraction
$\boldsymbol{P}_v$	Adjusted probability vector
$P_{v,j}$	Adjusted probability for ω_j
$E_v(\cdot)$	Cross-entropy function
log(·)	Base 10 logarithm

N	Total number of instances in the training set
E_v	Mean error
ξ_i	Training parameter
η_i	Training parameter
σ	Standard deviation
$\cdot^{\circ}$	Degree
θ	Angle
$H(P)$	Entropy of predictions
$F(\cdot)$	Empirical cumulative distribution function
$I(X_i \leq x)$	Indicator function

ACKNOWLEDGEMENTS

I express my deepest gratitude to my supervisor, Dr. Homayoun Najjaran, whose unwavering support and genuine care have significantly impacted my academic and personal growth. His belief in my potential has encouraged me to bring out the best in myself, and for that, I will always be thankful. A heartfelt thank you to the ACIS lab members who have shared their experiences with me, and I feel incredibly fortunate to have met and learned from them. Special appreciation goes to Dr. Kashish Gupta, whose kindness and insightful research discussions have been invaluable in shaping my early research steps.

To my dearest friends, you have been my pillars of strength, and I feel truly blessed to have you in my life. I sincerely thank Rawan Asfour, Yara Marghani, and Hanaa Allehaibi for their support to complete this thesis. My gratitude extends to my family, especially my sister Mariam, for their unconditional love and tremendous support throughout my life journey; without them, reaching this capacity of knowledge would not have been possible. Lastly, finding words to express my gratitude to my late mother, Khadija, is challenging. The values she instilled in me have been the primary motivations to pursue leaving my impactful mark on the world. All such success, both former and future, I channel to her.

DEDICATION

I dedicate this research to my mother.

Chapter 1

Introduction

Deep learning (DL) has significantly transformed the landscape of artificial intelligence (AI), ushering in a paradigm where computers learn and make decisions without explicit programming. At the heart of DL lies deep neural networks (DNNs), designed to mimic how the human brain functions and analyzes information. These networks, composed of interconnected layers of nodes or neurons, learn hierarchical features from data. Within AI, computer vision (CV) focuses on empowering computers to interpret visual information from images and videos. Convolutional neural networks (CNNs), a subtype of DNNs, have sparked a new era in CV with their distinct ability to learn from raw data, capturing essential characteristics like colors and edges and identifying patterns, features, and information hierarchies in images. For their extraordinary performance, CNNs have been widely adopted in various applications, spanning speech recognition [1], face

recognition [2], image classification [3], and diverse domains like healthcare [4], self-driving cars [5], and agriculture [6].

Nonetheless, despite the outstanding capabilities of CNNs, their performance diminishes in the face of high uncertainty. Uncertainty stems either from the model lack of knowledge (epistemic uncertainty) or inherent variations in data (aleatoric uncertainty). When uncertainty arises, typical machine learning (ML) classifiers fail to recognize it yet still provide predictions, posing a crucial dilemma of when to trust the model predictions. The ignorance of models about the correctness of the predictions provided restricts their applicability in scenarios where making precise decisions is crucial, and errors cannot be tolerated. For instance, in healthcare, precise disease classification from medical images, such as X-rays and MRIs, plays a crucial role in early medical intervention, as misclassification could jeopardize a patient's health. One approach to tackle this issue is to expose ML models to a massive variety of data with diverse randomness and variations. However, this is impractical.

A more practical approach involves empowering networks to quantify their uncertainty, allowing them to indicate instances of uncertainty for potential human intervention [7]. This establishes a collaborative human-AI model, relying on human intelligence at points of uncertainty in the AI model. Researchers have been keen on enabling DNNs to quantify uncertainty in predictions. Various methods, such as Bayesian learning and non-Bayesian approaches (discussed in Section 2.5), have been developed to enable models to estimate uncertainty. Nevertheless, some approaches encounter scalability issues, while others prove time-consuming in training and inference, rendering them unsuitable for real-time applications. Researchers have recently pursued an

alternative path that focuses on applying the principles of evidence theory, such as the Dempster-Shafer (D-S) theory in DNNs. This theory allows ML models to reason under uncertainty, making predictions and quantifying uncertainty effectively. Notably, this avenue of research has demonstrated outstanding performance in comparison to other methods. Consequently, this research exploits the concepts of evidential learning to present an evidential deep learning classifier with an added capability for uncertainty quantification (UQ).

1.1 Problem Statement

Assigning probabilities by a network to data classes is not necessarily indicative of the model confidence in the accuracy of these classes. The softmax layer in CNNs compels networks to compress computed features into a probability distribution between 0 and 1. Even when the extracted features do not strongly support a particular class, the model still generates a class probability within this range. This can lead to the assignment of high probabilities to wrong classes despite the underlying uncertainty represented in the computed features. To address this challenge, models should be empowered to quantify their uncertainty regarding the correct classification, enhancing their reliability, particularly in high-stakes applications. Consequently, this research aims to introduce a distance-based evidential deep learning classifier, d-EDL, equipped with an integrated ability to estimate its uncertainty. This is achieved by replacing the last two layers in typical CNN classifiers: the fully connected layer and the softmax layer with designed decision-making layers for more informed decisions.

1.2 Objectives

This study aims to achieve several key objectives:

- To develop a classifier that can effectively quantify its uncertainty when it cannot accurately determine the right classification for the input data. This is achieved by innovatively designing decision-making layers to replace the decision layers in conventional CNN classifiers.
- To provide a comprehensive method for converting feature vectors into BPAs and then merging these BPAs to form more informed decisions.
- To present an end-to-end training approach for the proposed classifier, facilitating the joint optimization of all network parameters, including those in the newly designed layers.
- To design evaluation experiments, each involves a distinct uncertainty leading factor to gauge the model ability to quantify ignorance rather than misclassification.
- To conduct an in-depth analysis of the model performance, provide insights into the behavior of the d-EDL classifier in the experiments, and propose potential avenues for future enhancements.

1.3 Contributions

This research makes significant contributions, outlined as follows:

- Introducing novel decision-making network layers that replace the softmax layer in CNNs, enabling networks to estimate uncertainty when faced with inputs that challenge confident classification.
- Presenting a detailed implementation of the architecture, training, and evaluation processes, which can serve as a guide for precise replication and inspiration for future research endeavors.
- Provide experimental designs for evaluations, specifically tailored to assess the capacity of a network to estimate uncertainty for unfamiliar data, incorporating factors that notably impact the network performance.
- Drawing insightful conclusions through analytical comparisons between the findings of this study and related work.
- Addressing the strengths and weaknesses of the proposed architecture, offering valuable guidance for shaping future research directions.

1.4 Thesis Outline

- **Chapter 1** begins by giving a brief introduction about the research topic, followed by stating the research problem, and outlining the research objectives and contributions. It also provides an overview of the structure of the thesis report.
- **Chapter 2** offers a concise background on essential concepts related to this study, and reviews popular methods in the literature for uncertainty quantification.
- **Chapter 3** discusses the methodology and details the steps taken to construct and train d-EDL.
- **Chapter 4** provides information about the datasets, implementation specifics, experimental processes, and evaluation metrics utilized in this study.
- **Chapter 5** presents and analyzes the research findings, offering valuable insights.
- **Chapter 6** concludes the work, emphasizing the significant outcomes, highlighting the impacts of the research, in addition to acknowledging study limitations, and suggesting avenues for future research.

Chapter 2

Background and Literature Review

The previous chapter discussed an introduction to the thesis topic alongside research problems, objectives, and contributions. To understand the work and the contributions of this thesis, this chapter begins by providing the necessary background on the essential concepts related to this study: probability theory, D-S theory, CNNs, and uncertainty. Later, it reviews the most popular UQ methods in the literature.

2.1 Probability Theory

Probability theory [8], [9], [10] is a mathematical discipline that explores the likelihoods associated with random events. Random phenomena inherently involve multiple potential outcomes, and probability theory employs various formal concepts to characterize the chances of specific outcomes. The fundamental concepts in probability theory related to set theory are briefly explained in the following subsections.

2.1.1 Random experiments, sample space, and events

In probability theory, a random experiment refers to a procedure, action, or process that, when carried out, leads to an outcome that is not precisely predictable or known beforehand. The term "random" implies that there is an element of uncertainty or unpredictability associated with the outcome of the experiment. In other words, the specific result or outcome of the experiment is not predetermined and can vary each time the experiment is conducted. The sample space, denoted by S, is a set containing all possible outcomes of a random experiment. For example, rolling a fair six-sided die can be considered a random experiment, with the sample space being $\{1, 2, 3, 4, 5, 6\}$. A subset of S is called an event, and it represents a collection of outcomes that share a common characteristic. An event is denoted by E, and $E \subseteq S$. For instance, the event of rolling a number that is even on the die is the subset $\{2, 4, 6\}$. The three fundamental axioms of probability are:

(1) Non-negativity: $P(A) \geq 0$

(2) Normalization: $P(S) = 1$

(3) Additivity: $P(A \cup B) = P(A) + P(B)$

where A and B are two events, and $\cup$ denotes the union symbol. The complement of event A is denoted as A^C and its probability is computed by:

$$P(A^c) = 1 - P(A) \tag{2.1}$$

2.1.2 Union and intersection

The union of two events, denoted $A \cup B$, comprises all the outcomes in either event A or event B or the two events together. The probability of $A \cup B$ reflects the occurrence of A or B, which is found by adding the individual probabilities of the events minus their intersection. The subtraction of their intersection is to avoid double-counting the overlapping outcomes. The probability of the union is expressed as:

$$P(A \cup B) = P(A) + P(B) - P(A \cap B) \tag{2.2}$$

The intersection of two events, denoted $A \cap B$, consists of the outcomes that belong to both A and B. $P(A \cap B)$ represents the probability of A and B to happen.

2.1.3 Independence and mutual exclusiveness

A and B are independent events only when the occurrence of one event does not influence the occurrence of the other event. The probability of two independent events is mathematically given as:

$$P(A \cap B) = P(A) \times P(B) \tag{2.3}$$

A and B are mutually exclusive if they cannot happen simultaneously, thus $A \cap B = \emptyset$. The probability of two mutually exclusive events is:

$$P(A \cap B) = 0 \tag{2.4}$$

2.1.4 Conditional probability

The probability of an event B happening given that event A has already happened is referred to as the conditional probability, and is computed by the subsequent equation:

$$P(B|A) = \frac{P(A \cap B)}{P(A)} \tag{2.5}$$

2.1.5 Bayes' theorem

Bayes' theorem is an essential concept in probability theory, providing a way to update probability estimates based on new evidence or information. The theorem is mathematically expressed as:

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)} \tag{2.6}$$

where $P(A)$ is the prior probability or initial belief of event A before considering any new evidence. $P(B|A)$ is the likelihood representing the probability of observing evidence B given that event A has occurred, quantifying how well the evidence is explained by the hypothesis A. $P(B)$ is the overall probability of observing evidence B, serving as a normalization factor to ensure that the updated probability is a valid probability distribution. The posterior probability is given by $P(A|B)$ which reflects the updated probability of event A given the new evidence event B.

2.2 Dempster-Shafer (D-S) Theory

Dempster-Shafer theory (D-S), alternatively known as evidence theory or belief theory, serves as a mathematical foundation for reasoning and decision-making under uncertainty, particularly when confronted with conflicting or incomplete evidence [11]. The theory was first initiated by Dempster [12] in 1967, which included the idea of upper and lower probabilities. Later, Shafer [13], in 1976, extended and modified the theory by replacing the upper and lower probabilities with the concept of degree of belief. This theory is perceived as the generalization of probability theory, as probabilities can be assigned to sets of events rather than just to singleton events. Therefore, if the evidence is adequate to assign probabilities only to single events, D-S converges to the traditional probability theory.

2.2.1. Frame of discernment (Ω)

The frame of discernment $\Omega = \{\omega_1, \omega_2, \dots\dots, \omega_M\}$ is an exhaustive set of all possible mutually exclusive hypotheses or events. In the traditional probability theory, if there are M distinct propositions, the occurrences of these events are presented by assigning probabilities $\{P_1, P_2, \dots\dots, P_M\}$ to these events in Ω such that $\sum_{i=1}^{M} P_i = 1$. On the other hand, D-S theory extends the probability assignments to subsets of Ω besides the singleton classes. It allows for assigning probabilities to the sets in the power set 2^{Ω} of the frame of discernment, while maintaining the constraint $\sum_{i=1}^{|2^{\Omega}|} P_i = 1$. Those

probabilities quantify the degrees of belief about the truthness of the propositions they represent.

2.2.2 Basic probability assignment (BPA)

The basic probability assignment (BPA) or the mass function m is a mapping function that maps every element A of 2^{Ω} to a probability value between [0,1]. The assignment of the belief functions is according to an evidence source, and the summation of the beliefs is equal to 1. In case of multiple evidence sources, each source assigns its own belief masses that sum up to 1. To combine the mass functions from different sources, D-S's rule of combination can be applied (see section 2.2.7). The axioms of the D-S theory are expressed mathematically as follows:

(1) $0 \leq m(A) \leq 1 \quad \forall A \subseteq 2^{\Omega}$

(2) $m(\emptyset) = 0$

(3) $\sum_{A \subseteq 2^{\Omega}} m(A) = 1$

Any element of the power set with $m(A) > 0$ is called a focal element.

2.2.3 Belief function (Bel)

A belief function represents how much the evidence at hand supports a given hypothesis. $Bel(A)$ quantifies the degree of belief that hypothesis A is the right answer. $Bel(A)$ can be found by summing up the mass functions of all subsets $m(B)$ of the set of interest A. Hence, the belief can be found by:

$$Bel(A) = \sum_{B \subseteq A} m(B) \tag{2.7}$$

2.2.4 Plausibility function (Pl)

A plausibility function for hypothesis A is computed by adding BPAs of sets B that intersect with the set of interest A. $Pl(A)$ represents the amount of belief reflecting the plausibility of hypothesis A, and it can be computed by:

$$Pl(A) = \sum_{B \cap A = \emptyset} m(B) \tag{2.8}$$

2.2.5 Uncertainty interval

The uncertainty interval is the area bounded by the belief and the plausibility within which the true probability might lie, as depicted in Fig. 2.1. This area can be found by:

$$Uncertainty\ interval(A) = Pl(A) - Bel(A) \tag{2.9}$$

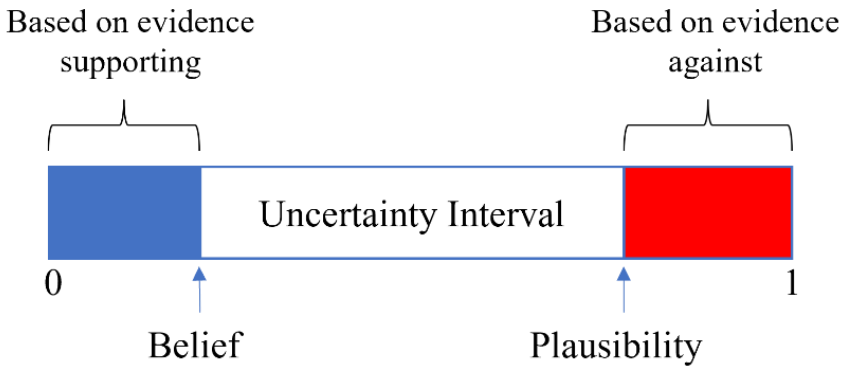


Figure 2.1: Uncertainty interval.

2.2.6 Pignistic probability

Pignistic transformation [14] is a method used to convert belief functions into a probability function for decision-making. In D-S theory, a belief function assigns masses to subsets of the sample space. The pignistic transformation offers a way to distribute these masses across individual events or hypotheses within the subsets. It is mathematically defined as:

$$BetP(A) = \sum_{A \subseteq B} \frac{m(B)}{|B|} \tag{2.10}$$

where $|B|$ is the cardinality of $B \subseteq \Omega$.

2.2.7 Dempster-Shafer's rule of combination

To merge the mass functions from two evidence sources, e.g., m_1 and m_2, D-S's rule of combination can be applied. The idea is to multiply BPAs of two sets whose intersect is the set of interest A, then apply normalization. The rule is mathematically expressed as follows:

$$m_c(A) = \frac{\sum_{B \cap C = A} m_1(B) m_2(C)}{\sum_{B \cap C = \emptyset} m_1(B) m_2(C)} \qquad A \neq \emptyset \tag{2.11}$$

2.3 Convolutional Neural Networks (CNNs)

CNNs have revolutionized the landscape of visual content processing, with Yann LeCun and colleagues introducing an early version, LeNet-5, in 1988 [15]. The LeNet-5 architecture was a groundbreaking design that was particularly created for recognition tasks, which later served as the basis for modern CNNs. In fact, the architecture of CNNs imitates the visual systems of human beings, including processing information in the neural system, from which the name “neural networks” was derived. The term “convolutional” was influenced by the primary operation in CNNs which is convolution. Over the years, researchers have designed numerous CNN architectures to achieve different tasks. The network architectures vary in their layer types, layer organization, and layer connections, as well as the depth and width of the network. This variety impacts how a network learns characteristics from input data and reflects the continuous innovation and research in deep learning. Some of the most popular CNNs in image classification are ResNet, VGG, and GoogleNet, and more CNN variants are discussed in detail in the review papers by [16] and [17].

Illustrated in Fig. 2.2 is a cat and dog deep learning classifier comprising interconnected layers of artificial neurons represented as blue circles. These neurons serve as computational units collectively processing the input image and generating a probability distribution as the network output. Each computational unit incorporates an activation function, introducing non-linearity to the network. The connections between layers, depicted by lines, signify the interactions among neurons in different layers, with

each connection possessing an associated weight influencing its significance. The orange circles represent biases, constants added to a neuron input to adjust the activation value. A neuron applies an activation function to its weighted and summed input, with the bias contributing to the final output, as shown in Fig. 2.3.

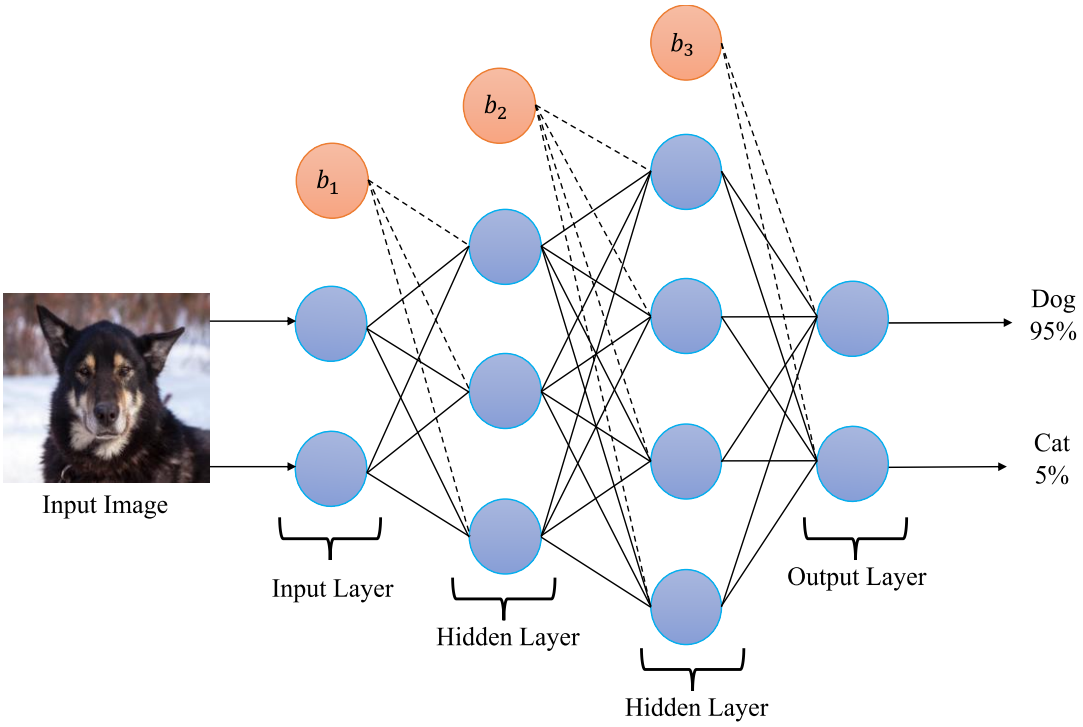


Figure 2.2: Example of a simple architecture of a deep learning classifier for a cat and dog dataset.

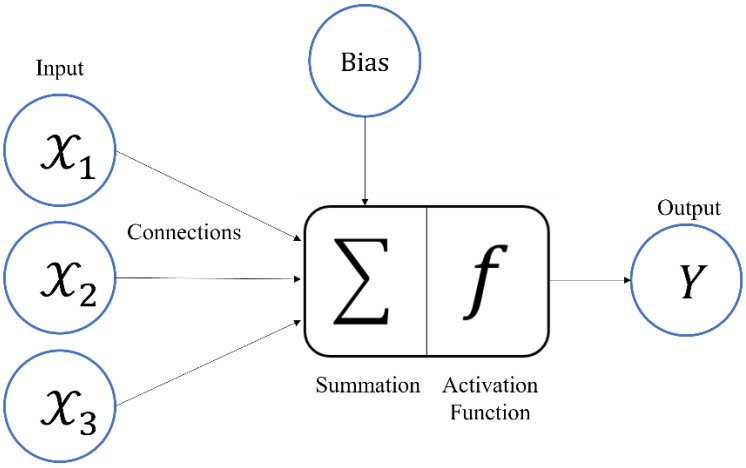


Figure 2.3: Visual demonstration of how a neuron processes input, generating an output activation.

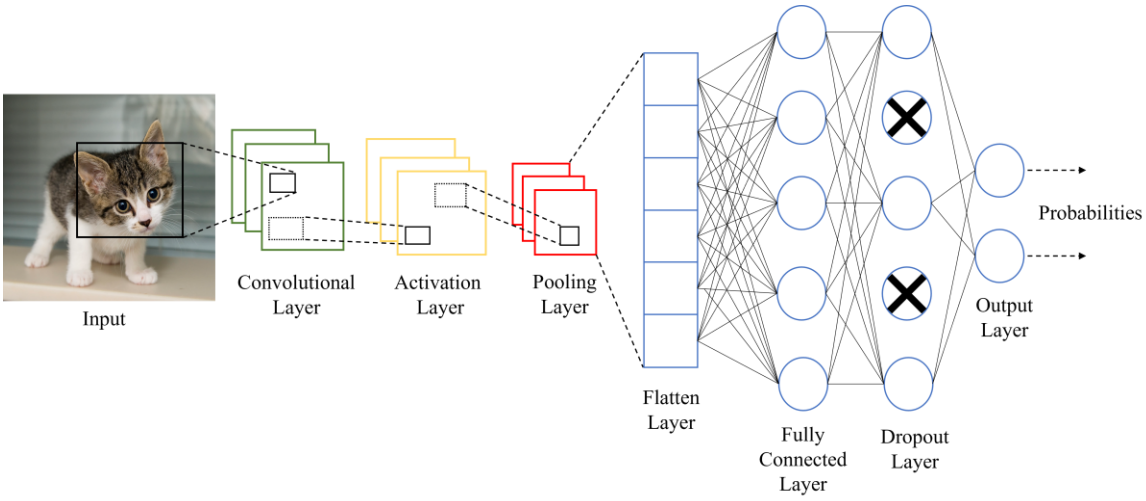


Figure 2.4: Example of an architecture of a simple CNN classifier showing the most popular layers found in CNNs.

Each network layer contributes uniquely to achieving the ultimate goal of the model, and the commonly used CNN layers are depicted in Fig. 2.4. Their roles are outlined as follows:

- An input layer is the initial layer in CNNs and is responsible for receiving raw data, such as images, and forwarding it to the network.
- Convolutional layers are the main layers in CNNs where multiple filters or kernels—small-sized windows are employed, sliding and convolving over the input to generate feature maps. The convolution operation consists of element-wise multiplication of the kernel with a segment of the input, then summing the individual values resulting from the multiplication (as depicted in Fig. 2.5). The primary objective of convolutional layers is to extract valuable patterns and features from the input, such as shapes, edges, etc.

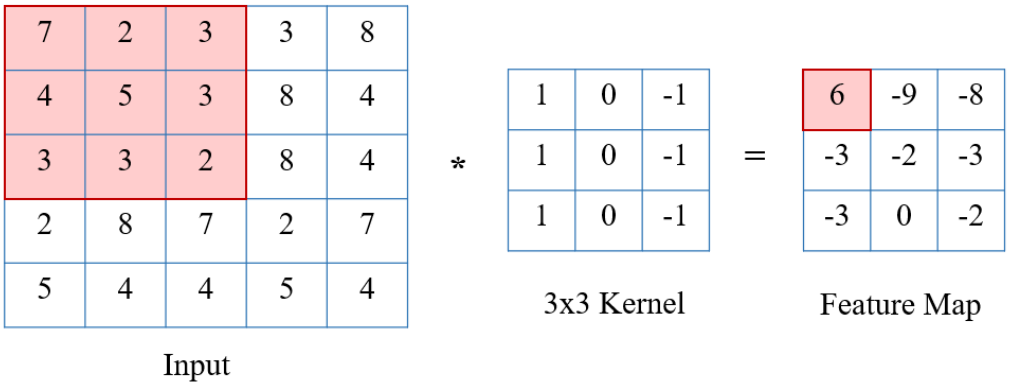


Figure 2.5: Visual illustration of a convolutional operation between an input activation and a convolutional kernel.

- Pooling layers come after convolutional layers to decrease the spatial dimensions of feature maps while maintaining the useful features captured by the convolutional layers, reducing computational complexity. The pooling operation involves selecting the maximum or average value from different input regions, as shown in Fig. 2.6.

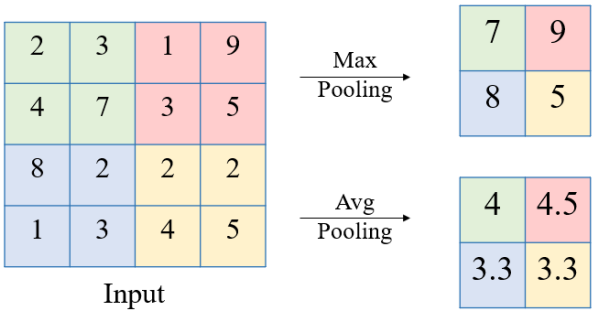


Figure 2.6: Visual demonstration of the pooling operation.

- Activation layers are often placed after convolutional and pooling layers to introduce non-linearity, enabling the capture of complex data patterns and relationships. Some of the commonly used activation functions are Rectified Linear Unit (ReLU), which generates zero for negative values, and Hyperbolic Tangent (Tanh), which squashes the output between -1 and 1.
- Fully connected layers learn high-level abstractions and global patterns, performing high-level reasoning in the network. Neurons in a fully connected layer are connected to every neuron in the preceding layer and the following layer. In CNN classifiers, there is usually a fully connected layer placed before the output layer, producing a set of scores or logits for every class.
- A dropout layer is a regularization technique employed in CNNs to avoid overfitting. This layer randomly deactivates a fraction of neurons during training based on a predefined dropping rate. For instance, a 0.5 rate means half of the neurons are deactivated, their outputs are not passed to the subsequent layer.

- An output layer converts received scores into a probability distribution over all classes, ensuring that the sum of predicted probabilities equals one. For multi-class classification, the softmax function is typically employed and can expressed as:

$$f(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{M} e^{x_j}} \tag{2.12}$$

where x_i is the input to the softmax function, and M is the total number of classes in a dataset.

Deep learning classifiers acquire knowledge by processing data through multiple network layers and computing the errors through a defined loss function. These errors, indicating how far off model predictions are from the actual values, are backpropagated to refine the network parameters. The trainable parameters encompass weights and biases, and throughout training, the model seeks optimal values for these parameters. Evaluating a trained model involves assessing its capacity to generalize to unseen data, ensuring accurate predictions on data not included in the training set. The most common method to train neural networks is Backpropagation [18], which operates in two phases to compute the gradient of the loss in terms of network parameters. In the first phase, input features are pushed forward through the network, computing the network output and, thereby, the associated loss with respect to the current network parameters. In the following phase, the gradients of the training loss concerning the weights and biases are sent backward from the output layer to the input, updating the values of the network parameters.

2.4 Uncertainty

Generally, the term “uncertainty” can refer to doubt, unpredictability, hesitancy, or indeterminacy regarding something, however, in the context of ML, uncertainty is mostly used to describe the lack of confidence or precision in the predictions of a model [19]. In other words, it reflects ignorance of the model about the true classification of the input sample, thus the correctness of the predictions provided. The probabilities assigned to data labels can be frequently misinterpreted as model confidence [20]. For example, when a model assigns a probability of 0.9 to a certain class, it might be interpreted as the model having 90% confidence in the correctness of that class as the true label. However, consider a counterexample where a cat and dog classifier assigns an 85% probability to a dog class for an input cow image, expressing high confidence in the wrong class. Therefore, it is important to avoid conflating predictions with confidence, and the development of reliable uncertainty estimation techniques is highly needed. In ML, uncertainty can manifest in two forms: aleatoric and epistemic uncertainty [21], [22], [23].

Aleatoric uncertainty, also referred to as statistical uncertainty, pertains to the inherent randomness or variability in experimental outcomes or the data. This type of uncertainty is irreducible; increasing the size of the training data cannot reduce this uncertainty because the randomness is backed in the system generating the data which has a factor that causes randomness in the produced outcomes. Epistemic uncertainty, also referred to as systematic uncertainty, stems from the model lack of knowledge and

inadequate learning of informative features, hindering its precise differentiation between classes. Unlike the aleatoric uncertainty, epistemic uncertainty can be potentially reduced by exposing the model to more training data.

2.5 Related Work

2.5.1 Bayesian learning (BL)

When talking about uncertainty estimation in neural networks (NNs), Bayesian neural networks (BNNs) [24], [25], [26] comes at the forefront. BNNs are probabilistic models where a distribution is placed over the parameters of these models. In standard NNs, there is an assumption that weights have a fixed true value, and the observed data is a random variable. In the context of probabilistic models, the model parameters are instead treated as random variables. Thus, the objective of the training phase is to use the training data to learn the posterior distribution of the model parameters. In most cases, the computation of the posterior distribution is infeasible due to the intractable integration computation of the denominator in the original Bayes' formula. Hence, rather than direct computations of the posterior, several Bayesian inference techniques to approximate the posterior have been developed over the years, some of which are discussed in the subsequent paragraphs.

One of the oldest techniques for posterior approximation is the Laplace approximation (LA) [27], [28], [29] which involves finding the mode maximum a posteriori (MAP) of the posterior distribution and then computing the Hessian matrix of the negative log-posterior at that mode. LA assumes a multivariate Gaussian distribution

centered at the mode with the Hessian matrix as the covariance matrix. The computation of the Hessian matrix may be impractical for large networks; therefore the overall performance might experience significant degradation [30]. Instead, the Gauss-Newton (GGN) matrix [31] is used to efficiently approximate the Hessian matrix, reducing the expensive computational cost.

Linearized Laplace approximation (LLA) [32], [33] is a variant of LA that includes additional simplifications yielding more tractable computations. LLA involves a step further by incorporating a linearization step. After obtaining the MAP estimate, it linearizes the log-posterior around this point, considering the posterior as a Gaussian distribution with the first-order Taylor expansion. This linearization can capture more information about the shape of the posterior distribution, especially in regions away from the mode. LLA exhibits greater performance than LA when coupled with GGN on various UQ tasks [33]. Nevertheless, it is challenging to pinpoint the approximation errors, significantly compromising the reliability of the learning outcomes [34].

Markov Chain Monte Carlo (MCMC) [35], [36] is a sampling-based technique leveraging Markov chains to estimate integrals and produce a sequence of posterior samples from the parameter space. The generated Markov chain converges to the posterior distribution, where the samples are used to approximate the posterior and estimate its properties. The chain begins by selecting a model from a predefined distribution of model parameters and suggesting a new model conditional on the current one. An acceptance criterion is calculated to accept or reject the suggested model. The downside of typical MCMC algorithms is a computation over the entire dataset is

required in each iteration, resulting in expensive computational costs for large datasets which hinders their use for big data analytics [37], [38].

Metropolis-Hastings (MH) algorithm [39] is one of the most popular MCMC algorithms, which performs random walks involving stochastic changes leading to convergence. Examples of scalable MH techniques in the literature are minibatch MH [40], scalable Metropolis-Hastings (SMH) [41], and Differential Privacy-Fast Metropolis-Hastings (DP-Fast MH) [42]. Despite the spectacular performances of scalable MHs, MH still encounters difficulties in high-dimensional spaces due to its random-walk nature, resulting in slow convergence, especially when navigating complex distributions with correlated parameters. Inefficient exploration becomes prominent when the current state is distant from regions of high posterior density, leading to low acceptance rates for proposed samples and impeding the efficiency of the algorithm [43].

Hamiltonian Monte Carlo (HMC) algorithm [24], [44], [45] is another variant of MCMC that applies Hamiltonian dynamics and physics-inspired principles to perform sampling from probability distributions. HMC contributes to improving the efficiency of MH by utilizing the gradient of the logarithm of the posterior distribution. This directs the Markov chain towards regions having the majority of concentrated samples, high posterior density. As a result, a well-optimized HMC chain exhibits a substantially higher acceptance rate for proposals compared to the conventional MH algorithm [46]. Hence, HMC can theoretically find the true posterior distribution [47]. Nonetheless, HMC suffers from poor scaling ability [48], in addition to the convergence to the true posterior when model assumptions are not met cannot be theoretically guaranteed [49].

Stochastic Gradient Markov Chain Monte Carlo (SG-MCMC) [37], [50], [51] is a sampling method that combines stochastic gradient optimization with MCMC techniques for Bayesian inference in deep learning models. The process involves iteratively updating the model parameters using stochastic gradients derived from subsets of the training data. Therefore, SG-MCMC utilizes mini-batches of data, and empirical evidence suggests its significance in discovering weight parameters that lead to effective generalization in DNNs [52]. However, while SG-MCMC is theoretically expected to converge asymptotically to target distributions through a diminishing stepsize strategy, it faces a constrained estimation error within finite time periods [53], [54]. The practical challenge stems from reported successes in training DNNs within short durations, creating a mismatch with the theoretical expectations of SG-MCMC [55].

Variational Inference (VI) [56] is an optimization-based method that is an excellent substitute for MCMC methods when sampling is intractable and the posterior probability density functions cannot be calculated analytically [8]. VI introduces a tractable family of distributions and seeks the family member that minimizes the Kullback-Leibler divergence from the true posterior. By optimizing a set of variational parameters, VI transforms the intricate probabilistic inference problem into an optimization problem. This offers the benefits of MAP estimation [57], as well as facilitates seamless scalability to extensive datasets [58].

Bayes by Backpropagation (BBB) [59] and Probabilistic Backpropagation (PBP) [30] represent two prominent approaches in VI for BNNs. BBB assumes a factorized Gaussian distribution for each weight, treating them as independent random variables and optimizing the distribution parameters through variational inference. On the other hand,

PBP follows a two-phase structure, sequentially approximating intractable distributions generated by random weight treatments and updating the means and variances of the Gaussian posteriors. While both methods utilize factorized Gaussians, BBB captures uncertainty through weight variances, while PBP draws several samples from the posterior to estimate the uncertainty. Both approaches assume the posterior distribution as a factorized Gaussian; hence their performance might degrade in scenarios where learning the correlated uncertainties of the parameters is crucial [60].

There are methods crafted to account for posterior correlation by introducing structured approximation families. Matrix Gaussian Posteriors (MVG) [61] utilizes the parameter posterior distribution of a matrix-variate Gaussian [62] for explicit modeling of the covariance among the input and output dimensions within each layer, leading to an efficient representation of these correlations. Multiplicative Normalizing Flows (MNF) approach [63] captures correlations by reinterpreting multiplicative noise in NNs as auxiliary random variables, resulting in efficient incorporates normalizing flows, allowing for local reparameterizations and a tractable lower bound. Bayes by Hypernet (BbH) [64] captures complex correlations between the weights of NNs by utilizing hypernetworks [65] interpreted as implicit distributions to approximate the posterior distribution.

2.5.2 Non-Bayesian learning (NBL)

Ensemble methods [66], [67] quantify uncertainties by training multiple neural networks with distinct weights independently, followed by joint training using a shared loss function. Each network in the ensemble provides a different perspective on the data, and

uncertainty is estimated by analyzing the variability in predictions among the ensemble members. In a similar manner, Bootstrapping [68] involves creating several subsets of the training data through bootstrap resampling, where each subset is utilized to train a distinct model. These models, often based on the same learning algorithm, then contribute collectively to the final predictions through averaging for regression or voting for classification. Both Ensemble and Bootstrapping methods offer avenues for uncertainty estimation without substantial architectural modifications, with Ensemble exhibiting enhanced robustness to posterior domain shifts, as per [69]. The drawback of such methods is the time delay induced by several forward passes through the different networks involved.

Monte Carlo (MC) Dropout [70], [71] can effectively achieve the excellence of Bayesian methods for uncertainty estimation with much simpler network architectures. It accomplishes that by inserting a dropout layer, usually employed to avoid overfitting, after every network layer with weights, turning off some neurons based on the dropout rate. At the test time, the added dropout layers are not removed but rather used to introduce stochasticity to the trained network. The network is run multiple times and the different predictions obtained are averaged and the variance in predictions is used to estimate the uncertainty. The selection of suitable rates for multiple dropout layers is based on experimenting with different rates and evaluating the network performance accordingly.

Several studies attempted to find the optimal dropout probability value by computing the gradient with respect to the dropout parameter. Concrete Dropout [72] solves this optimization problem by using a relaxation of the Bernoulli Distribution, Variational

Dropout [73] by employing a Gaussian approximation, and Learnable Bernoulli Dropout (LBD) [74] by utilizing an Augment-REINFORCE-Merge estimator of the Bernoulli gradient. Dropout Regulation (DR) [75] is a non-gradient-based technique that regulates the rates based on the performance gap between the training data and the validation data. Therefore, a higher dropout probability is needed when the model is overfitting for a stronger regularization.

2.5.3 Evidential learning (EL)

Evidential learning is an approach to uncertainty quantification that goes beyond traditional probabilistic frameworks, involving modeling uncertainty using belief functions from Dempster-Shafer theory. Instead of relying solely on probabilities, evidential learning considers multiple pieces of evidence and their associated beliefs. This framework allows for a more nuanced representation of uncertainty, especially in situations with limited or conflicting information. The evidential classifier in [76] formulates evidence by employing a distance-based metric to measure the similarities between input data and reference patterns having a certain degree of membership to every class. Later, the rule of combination by D-S is used to combine the evidence supporting each class and compute the overall uncertainty. The output is a set of probabilities assigned to data classes, along with a designated probability for uncertainty. The probabilities for all classes, including the uncertainty class, sum up to 1. A set-valued classifier [77] is an extension of [76] that integrates a CNN feature extractor with the evidential layers from [76] and an expected utility layer for decision-making. In set-

valued classification, an observation is assigned to a non-empty subset of the power set, allowing for more precise uncertainty quantification.

Evidential Deep Learning (EDL) [78] constitutes evidence by employing a Dirichlet distribution on class probabilities, viewing predictions as subjective opinions, and dynamically learning the parameters of the distribution during training. EDL generates a set of probabilities summing up to one, accompanied by a separate uncertainty variable ranging from zero to one. For this classifier, a predefined uncertainty threshold needs to be set, and predictions accompanied by uncertainty above the threshold are considered uncertain. Selecting the threshold is subjective to the model user, and this can be a critical issue in some applications if the user lacks expertise in properly choosing the suitable threshold value. Modified-EDL (m-EDL) [79] improves upon this by extending the output probability distribution to include an additional class for uncertainty. m-EDL automatically learns evidence for uncertainty during training, eliminating the need for the user opinion in selecting the threshold. Moreover, this enhancement is particularly beneficial to the model flexibility in learning outliers that cannot be accurately classified during training, thus safely assigning them to the uncertainty class.

Chapter 3

Methodology

The preceding two chapters discussed the research topic, stated the research problems and objectives, shared the necessary background to understand this chapter, and reviewed the related work. This chapter extends the discussion and outlines the step-by-step process to construct and implement the proposed classifier.

The proposed classifier, d-EDL, comprises two distinct parts: the CNN feature extractor (FE) and the decision-maker, as shown in Fig 3.1. The decision-maker replaces the final two layers in conventional CNN classifiers: the fully connected and the softmax layers. The feature extractor extracts meaningful features from input data instances and sends them to the decision-maker. The decision-maker converts the input features into BPAs and applies rules to combine these BPAs. The conversion involves measuring the proximity of input data to data representatives. For each data label, a certain number of data representatives are formulated using a clustering method. The output of a network is a probability distribution extended to include an uncertainty class, Ω, whose probability value reflects how much the model does not know the right classification.

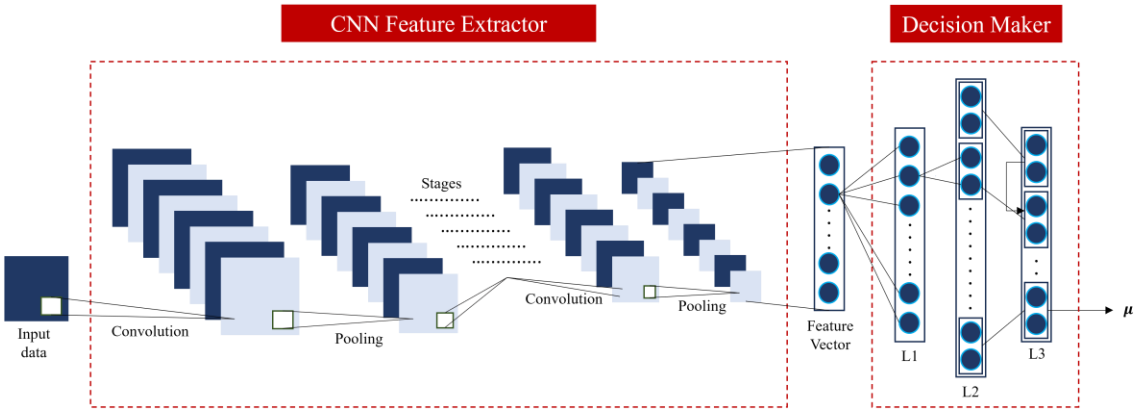


Figure 3.1: The architecture of the d-EDL classifier consisting of a CNN feature extractor and decision-making layers (L1, L2, and L3).

3.1 Feature Extraction

Since CNNs have gained a well-deserved reputation for their great ability to extract invaluable features from raw data, this influenced the decision to leverage them in this study rather than dealing with raw data to compute BPAs. To convert a CNN classifier into an FE, the last two layers (fully connected and softmax layers) were removed. A flatten layer was added to the end of FE, ensuring that the output is a one-dimensional vector of features. A feature vector is expressed as:

$$\boldsymbol{z}_i = (z_1, z_2, \dots\dots\dots, z_b)^T \tag{3.1}$$

where b represents the length of the feature vector and T is the symbol for transpose. $\boldsymbol{z}_i$ is fed into the next part of d-EDL, the decision-maker, and Sections 3.2 to 3.4 explore it in detail.

3.2 Representative Patterns

The decision-making process relies on measuring the similarities between input data and dataset representatives, with differences serving as markers of ambiguity. It is presumed that each test sample x is close to some training samples $x_i \in X$ by some distance d^i. Computing the distances between x and each x_i is computationally expensive. Even applying a search algorithm, such as k-nearest neighbors (k-NN), to find the nearest x_i to x is still costly [76]. In fact, the computational complexity of such an algorithm increases with the size of the dataset. Hence, a set of n data representatives is derived from the dataset and referred to as prototypes $\{\boldsymbol{p}_1, \boldsymbol{p}_2, \dots\dots, \boldsymbol{p}_n\}$, as described in [76], [77]. The proximity of x to each $\boldsymbol{p}_i$ serves as evidence about the true class of x. Fig. 3.2 displays the steps involved in computing the prototypes.

First, a conventional CNN classifier was trained on the dataset until satisfactory performance was obtained. Then, the classifier was converted into an FE, and only the training set was passed through the FE, generating feature vectors. In the following step, $\boldsymbol{z}_i$ was aggregated, constituting a new dataset $Z = \{\boldsymbol{z}_1, \boldsymbol{z}_2, \dots\dots\}$ having the same number of data instances as the original training set X. The instances in training set X were passed in order, hence the aggregated Z kept the same order of X. Therefore, training label set Y was used to split Z into M groups where each group contained all $\boldsymbol{z}_i$ belonging to the same class ω_j, where $\omega_j \in \Omega = \{\omega_1, \omega_2, \dots\dots\dots, \omega_M\}$. Each ω_j has a certain number of representatives k_i, and the total number of representatives for the entire dataset is:

$$n = \sum_{i=1}^{M} k_i \tag{3.2}$$

At this stage, the feature vectors had been separated based on their true class labels, k-means clustering algorithm [80] was applied to each of the M groups separately to form a total of n clusters. The n cluster centroids, denoted as $\boldsymbol{c}_i$, are vectors having the same length as $\boldsymbol{z}_i$, and will serve as initial vectors for the representatives. To determine the optimal number of clusters k_i for every ω_j, the clustering algorithm was run in a loop in conjunction with the elbow method [81] with different values of k. For every value of k, the elbow method calculated the within-cluster sum of squares (WCSS), measuring the sum of the squared distances between data points and their cluster centroids. The WCSS is mathematically defined by:

$$WCSS = \sum_{i=1}^{k_j} \left\| \boldsymbol{z} - \boldsymbol{c}_{ij} \right\|^2 \tag{3.3}$$

where $\boldsymbol{z}$ denotes the feature vector of an input data, and $\boldsymbol{c}_{ij}$ is the i^{th} centroid representing class ω_j.

In the plot of WCSS versus the number of clusters, as k increases, $WCSS$ decreases, and an elbow point is formed when the curve bends sharply. The k value on which the elbow point has occurred reflects the optimal number of class representatives. However, during implementation, it was observed that there was no clear elbow point for all classes. This was due to the nature of the data being clustered, the dataset used in this study is the MNIST dataset, which shows a high degree of similarity between images belonging to the same class. Therefore, the feature vectors exhibited high similarities;

hence, forming distinct clusters was quite challenging. Subsequently, it was decided to keep k fixed for all data labels.

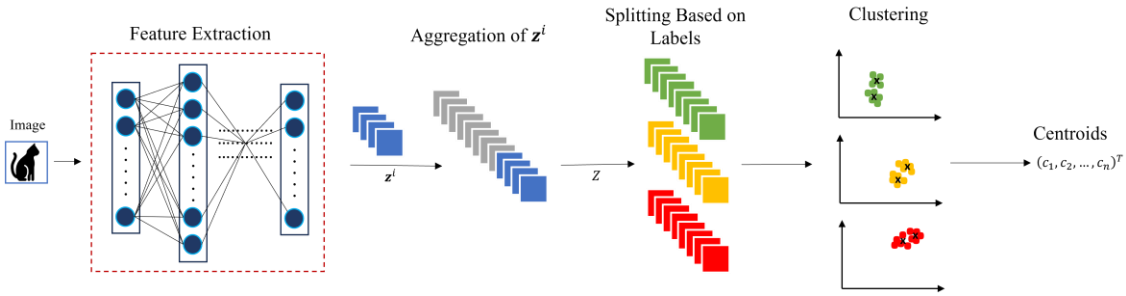


Figure 3.2: Illustration of the steps involved in finding data representatives for each data label.

3.3 Basic Probability Assignments (BPAs)

After finding data representatives, it was time to use these representatives to compute BPAs for $M+1$ classes $\{\omega_1, \omega_2, \ldots\ldots\ldots, \omega_{M+1}\}$ where ω_{M+1} denotes the BPA for Ω reflecting the model ignorance about the classification of x. The similarity between $\boldsymbol{z}$ and each $\boldsymbol{p}_i$ was found by employing the Euclidean distance metric, and it is mathematically expressed as:

$$d_i = \|\boldsymbol{z} - \boldsymbol{p}_i\| \tag{3.4}$$

The BPA reflecting the degree of belief committed to ω_j can be computed by:

$$s_i = \alpha_i \exp\left(-\gamma_i (d_i)^2\right) \tag{3.5}$$

where α_i and γ_i are positive values associated with each data representative, controlling the weight and spread of the decaying function, respectively. s_i is a monotonically

decreasing function, such that $s_i(0) = 1$ and $\lim_{d_i \to \infty} s_i(d_i) = 0$. If s_i reflects the degree of belief in ω_j to be the correct class and $0 \leq s_i \leq 1$, thus $1 - s_i$ is the disbelief in ω_i. The disbelief indicates that x could belong to any class $\omega_r \in \Omega$ but ω_j, or to $\omega_r \notin \Omega$. Each class representative was perceived as an evidence source E_i, and a BPA vector was formed from each representative and given as:

$$\boldsymbol{m}_i = \left(m_i(\omega_j), m_i(\Omega)\right)^T = (s_i, 1 - s_i)^T \tag{3.6}$$

where $\boldsymbol{m}_i$ is a vector of BPAs coming from E_i influencing the degree of belief about the correct label of x.

3.4 Combining BPAs

Two rules were applied to merge n generated $\boldsymbol{m}_i$ vectors, producing the d-EDL final output. Fig. 3.3 displays an example of the two rules to combine $\boldsymbol{m}_i$ from 9 evidence sources representing three labels $\{A, B, C\}$, with each class having three evidence sources. First, $\boldsymbol{m}_i$ from all E_i representing the same ω_j were merged, forming the overall model belief and disbelief in ω_j. Each E_i is an independent source of evidence, and the multiplication rule in the probability theory was iteratively applied to find the joint beliefs from each two sources as follows:

$$q_j(A) = m_i(A) \times m_r(A) \qquad A \in \{\omega_j, \Omega\} \tag{3.7}$$

Normalization was applied to ensure that the joint belief and disbelief for each label still fall within the range [0,1]. The normalization rule is:

$$q_j\_norm(A) = \frac{q_j(A)}{q_j(\omega_j) + q_j(\Omega)} \quad A \in \{\omega_j, \Omega\} \; and \; j = 1, 2, \ldots \ldots, M \tag{3.8}$$

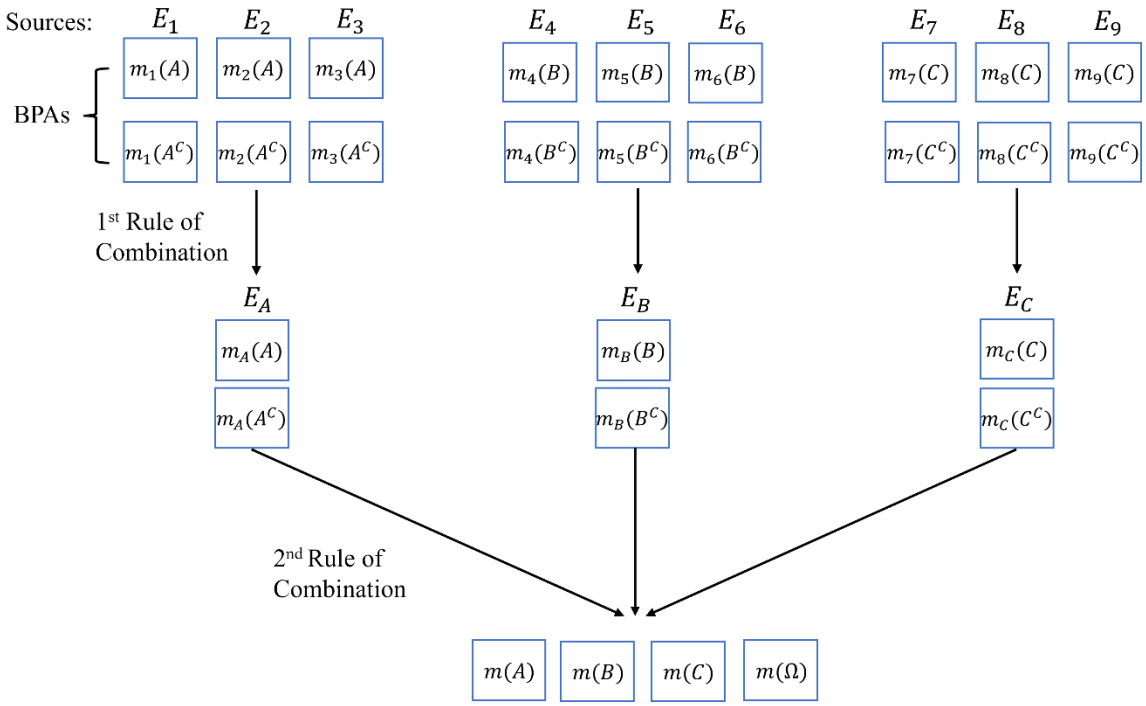


Figure 3.3: Illustration of the rules applied to combine BPAs for $\Omega = \{A, B, C\}$.

The first rule resulted in a reduced number of evidence sources to M. The second rule applied was inspired by Dempster-Shafer's rule of combination and given by:

$$\mu_j(\omega_j) = \sum_{r=1, r \neq j}^{M} q_j(\omega_j) \times q_r(\Omega) \qquad j = 1, 2, \ldots., M \tag{3.9}$$

For class Ω, the following formula was employed iteratively:

$$\mu_i(\Omega) = \mu_{i-1}(\Omega) \times q_i(\Omega) \quad i = 2, 3, \ldots \ldots, M \tag{3.10}$$

where $\mu_1(\Omega) = q_1(\Omega)$. Then the results were normalized by:

$$\mu(A) = \frac{\mu(A)}{\sum_{j=1}^{M} \mu(\omega_j) + \mu(\Omega)} \qquad A \in \{\omega_1, \ldots., \omega_M, \Omega\}$$

(3.11)

3.5 Parameter Learning

The proposed evidential classifier was trained by measuring the errors between predictions $\boldsymbol{\mu}$ and the ground truth vector $\boldsymbol{t}$. The error can be quantified by computing the cross-entropy function or any other loss functions typically used for CNN training. The output vector $\boldsymbol{\mu}$ is of length $M + 1$, and the true label vector has only M elements, $\boldsymbol{t} = (t_1, t_2, \dots\dots, t_M)^T$. Therefore, there was a need to convert $\boldsymbol{\mu}$ to have M elements to compute the accuracy and the loss. This was accomplished by distributing a fraction v of $\mu(\Omega)$ to every $\mu(\omega_j)$ for all $\omega_j \in \Omega$, thus distributing the uncertainty equally between classes. To achieve that, a new vector $\boldsymbol{P}_v = (P_{v,1}, P_{v,2}, \dots\dots, P_{v,M})^T$ was computed using the below vector transformation formula:

$$P_{v,j} = \mu(\omega_j) + (v \times \mu(\Omega)) \tag{3.12}$$

where $0 \leq v \leq 1$. $P_{0,j}$ denotes the belief, $P_{1,j}$ is plausibility, and $P_{1/M,j}$ represents the pignistic probability for class ω_j.

Pignistic probability in decision theory refers to the probability that a rational person gives to a choice when decision-making is required. This research considered the class with the highest pignistic probability to be the predicted class for x. The output error $E_v(x)$ for a given x and v can be measured by the cross-entropy function expressed below:

$$E_v(x) = -\sum_{j=1}^{M} t_j log\left(P_{v,j}\right) + (1 - t_j) \times log\left(1 - P_{v,j}\right) \tag{3.13}$$

The training aimed to find the parameter values that minimize the mean error E_v for a training set X consisting of N instances. The mean error was found by:

$$E_v = \frac{1}{N} \sum_{x \in X} E_v(x) \tag{3.14}$$

The parameters $\boldsymbol{p}_i$, α_i and γ_i were optimized during training to reduce E_v, under the constraints $0 < \alpha_i < 1$ and $\gamma_i > 0$. These constraints were applied during training by introducing new parameters ξ_i and η_i such that:

$$\alpha_i = \frac{1}{1 + \exp\left(-\xi_i\right)} \tag{3.15}$$

and

$$\gamma_i = (\eta_i)^2 \tag{3.16}$$

Chapter 4

Implementation Details and Experimental Setup

This chapter provides an overview of the datasets, discusses model architectures, and outlines the training methods employed. It then delves into the experiments designed to assess the classifier performance and shares the evaluation metrics considered in this study.

4.1 Datasets

The dataset utilized to train models in this study was the MNIST dataset [82], consisting of 28x28 grayscale images depicting handwritten digits from 0 to 9. This dataset is divided into a training set comprising 60,000 images and a test set of 10,000 images, each corresponding to a single digit. To evaluate the performance on OOD data, three

additional datasets were incorporated: Fashion MNIST [83], EMNIST [84], and KMNIST [85].

Fashion MNIST comprises 28x28 grayscale images showcasing various fashion items, categorized into 10 classes. Each class represents different types of clothing or accessories, such as t-shirts, trousers, dresses, and shoes. Similar to MNIST, Fashion MNIST includes 60,000 training images and 10,000 test images.

The EMNIST (Extended MNIST) Balanced dataset includes 47 classes, with each class corresponding to a different handwritten alphanumeric character. EMNIST shares the same 28x28 grayscale image format as MNIST, featuring 112,800 images in the training set and 18,800 in the test set.

KMNIST (Kuzishiji-MNIST) is another dataset containing 28x28 grayscale images with 10 classes representing handwritten characters written in Japanese. Similar to MNIST and Fashion MNIST, KMNIST consists of 60,000 and 10,000 images in the training and test sets, respectively.

All image datasets were normalized by dividing each pixel in an image by the maximum value a pixel can have, 255.0. This simple normalization technique aims to rescale the pixel values to fall within the range [0,1]. The MNIST training set was divided into 90% for training and 10% for validation. For the other datasets, only the test set from each was used to evaluate the model for OOD data.

4.2 Network Architectures and Training Details

4.2.1 d-EDL

The architecture of the proposed evidential model consists of 8 layers, as listed in Table 4.1. As depicted in Fig 3.1, d-EDL consists of a feature extractor and a decision maker. The features extractor contains two convolutional layers, two Max pooling layers, and ending with a flatten layer outputting a 1D feature vector. The decision-maker consists of three designed layers, each designated to perform a task. The decision-maker takes the 1D vector feature and applies a distance-based metric, measuring the similarities between data instances and prototypes constituting BPAs. Later, BPAs are combined in two steps, forming the final decision by the network containing $M + 1$ predictions for M classes and the uncertainty class.

The role, number of neurons, type of activations, and connections for each layer are as follows:

- The first layer (L1) consists of n units with activation function $s_i = \alpha_i \exp(-\gamma_i (d_i)^2)$ for $i = 1, 2, \dots\dots\dots, n$ to compute the BPAs. Each unit is fully connected to the previous layer, and the connection weights between the last layer in the feature extractor and L1 represent $\boldsymbol{p_i}$. In other words, the weights of the connections of each unit in L1 collectively form the vector $\boldsymbol{p_i}$. The weights were initialized to the cluster centroids found by the clustering algorithm and updated during training.

Table 4.1: d-EDL layers.

Layer	Filters/Neurons	Patch Size	Stride	Activation
Conv2D	20	5×5	1	ReLU
Max Pooling	–	2×2	2	–
Conv2D	50	5×5	1	ReLU
Max Pooling	–	2×2	2	–
Flatten	–	–	–	–
Custom Layer (L1)	n	–	–	–
Custom Layer (L2)	$2 \times n$	–	–	–
Custom Layer (L3)	$2 \times M$	–	–	–

- There are n modules in the second layer (L2), each has 2 units and the i^{th} module in this layer is only connected to the i^{th} unit in L1. All connection weights between L1 and L2 were set to 1. This layer performs the first rule of combination by applying Eq. (3.7) and normalizes with Eq. (3.8).
- For the third layer (L3), the received merged BPAs from L2 are combined again in this layer using Eq. (3.9) to Eq. (3.11). Therefore, this layer generates the final decision vector $\boldsymbol{\mu}$ of the network. The 3rd layer has M modules of 2 units, however each module is connected to two modules, except the first module. The i^{th} module in

L3 receives input from module $i-1$ lying in L3 and module i in L2. While the first module in L3 is only connected to the first module in L2. The inner connections and the connections between L2 and L3 have a fixed value equal to unity.

First, a standard CNN classifier with an architecture specified in Table 4.2 was trained only on the MNIST dataset, using the "Adam" optimizer, a learning rate of 0.001, and a batch size of 32. This classifier served solely the purpose of extracting feature vectors for data clustering and data representative formation. Hence, the learned weights in the CNN classifier were not transferred to the d-EDL classifier. Five variants of d-EDL, each having a different number of data representatives $n \in \{20, 40, 60, 80, 100\}$ were trained using the "Adam" optimizer with a 0.0005 learning rate and a batch size of 256.

4.2.2 Other models

To evaluate the performance of the proposed network as a classifier and uncertainty quantifier, three models were considered, LeNet (a Standard CNN), EDL [78], and m-EDL [79]. The details of network layers for the three models are presented in Table 4.2, Table 4.3, and Table 4.4, respectively. The Standard CNN is a typical classifier with the softmax activation applied to the last layer output, producing M probabilities. EDL is the state-of-the-art network outperforming all the other uncertainty quantification methods mentioned in Section 2.5. This network outputs M predictions for M classes, summing up to 1 in addition to a separate value for uncertainty ranging [0,1], indicating how much the model does not know the right answer. m-EDL is an improvement over EDL, and analogous to d-EDL, it generates $M+1$predictions, adding up to 1.

Table 4.2: CNN classifier layers.

Layer	Filters/Neurons	Patch Size	Stride	Activation
Conv2D	20	5 × 5	1	ReLU
Max Pooling	–	2 × 2	2	–
Conv2D	50	5 × 5	1	ReLU
Max Pooling	–	2 × 2	2	–
Flatten	–	–	–	–
Dense	500	–	–	ReLU
Dense	10	–	–	Softmax

Table 4.3: EDL layers.

Layer	Filters/Neurons	Patch Size	Stride	Activation
Conv2D	20	5 × 5	1	ReLU
Max Pooling	–	2 × 2	2	–
Conv2D	50	5 × 5	1	ReLU
Max Pooling	–	2 × 2	2	–
Flatten	–	–	–	–
Dense	500	–	–	ReLU
Dropout (0.5)	–	–	–	–
Dense	10	–	–	ReLU

Table 4.4: m-EDL layers.

Layer	Filters/Neurons	Patch Size	Stride	Activation
Conv2D	20	5×5	1	ReLU
Max Pooling	–	2×2	2	–
Conv2D	50	5×5	1	ReLU
Max Pooling	–	2×2	2	–
Flatten	–	–	–	–
Dense	500	–	–	ReLU
Dropout (0.2)	–	–	–	–
Dense	11	–	–	ReLU

The “Adam” optimizer was also used to train the three models on the MNIST dataset with a 0.001 learning rate. The batch size was 256 for CNN and EDL and 32 for m-EDL. The codes for the loss functions and the computations of the predictions and uncertainty from EDL and m-EDL outputs were taken from this GitHub source [86]. The annealing coefficient for EDL was set to $\min(1.0, \frac{ep}{10})$ as per the EDL paper where ep refers to the index of the training epoch. However, the annealing coefficient for m-EDL was set to 0.1 as it was noticed that the annealing coefficient set to $\min(1.0, \frac{ep}{10})$ led to poor training for the m-EDL network.

4.3 Experiments

The primary objective of this thesis is to design an evidential classifier capable of expressing its uncertainty when faced with ambiguous input data. To comprehensively evaluate the model proficiency in estimating uncertainty, three experiments were designed, each tailored to gauge the network response to a different source of uncertainty: noise, image rotation, and out-of-distribution data. The initial step is evaluating the model performance on the test set of the dataset that was trained on, ensuring effective categorization with minimal errors before exposing the model to unfamiliar data. Following the successful categorization of test images, the model becomes qualified for evaluation through the three experiments.

4.3.1 Experiment 1: noise

This experiment aims to evaluate the model performance in the presence of various levels of noise added to the MNIST test set. Noise can significantly degrade model performance, and this assessment provides insights into the model robustness under noisy conditions. White Gaussian noise, characterized by its standard deviation (σ), affects images by introducing random variations in pixel values. Higher σ implies a broader spread of these variations, leading to more noticeable distortion and making it challenging for models to correctly classify the images. As σ increases, models may struggle to distinguish signal from noise, resulting in decreased performance and increased uncertainty.

Gaussian noise was introduced with a mean of 0 and varying σ within the range [0,1]. The standard deviation allows for adding noise systematically to images, controlling the levels of noise introduced, thereby facilitating the analysis. For each σ value, 10 sets of noisy image datasets were generated by adding noise to the original images. The process of adding noise involves drawing random values from a normal distribution, thus each set has different variations of noise patterns. The model will be assessed on its average performance across these sets for each σ, gauging its capability to make accurate classifications and express uncertainty.

4.3.2 Experiment 2: image rotation

The evaluation against image rotation also focuses on understanding how well a model can maintain its predictive accuracy or quantify its uncertainty when exposed to varied degrees of rotation. The image rotation experiment involved systematically rotating the entire test set from the MNIST dataset with different angles within the range of $[0^{\circ},360^{\circ}]$, incremented by 30°. This experiment delves into studying the model performance under the impact of rotational transformations in a 2D plane, simulating real-life scenarios where images are captured from different angles for objects in a 3D space, thereby influencing the appearances of these objects in images. Since it is not feasible to include images captured from every combination of angles in a 3D space in the training data, models often struggle when deployed. Therefore, assessing their ability to express uncertainty in such situations where accurate classification becomes challenging is crucial.

4.3.3 Experiment 3: OOD data

Testing models against out-of-distribution data is essential, simulating scenarios where they encounter data that differs significantly from their training data. OOD testing helps assess how well the model can recognize and respond to unfamiliar patterns or classes not present in the training data. Models that can effectively estimate uncertainty when faced with OOD data are more reliable in real-world applications. Therefore, this experiment focuses on assessing the model performance in such situations, using test sets from three datasets: Fashion MNIST, EMNIST, and KMNIST. This experiment aims to assess the model capability to estimate uncertainty and explore whether this ability is influenced by the similarities between the training dataset and the data used for testing. The hypothesis is that data closely aligned with the training distribution might lead to lower uncertainty and a reduced detection for OOD data, as it closely resembles the patterns the model was initially trained on.

4.4 Evaluation Metrics

4.4.1 Accuracy

The assessment of prediction accuracy is a fundamental metric, quantifying the instances where the model correctly assigns input data to their true classes, providing an overall performance in making accurate predictions. However, in scenarios where the model struggles to make precise classifications, it is preferable for the model to effectively quantify its uncertainty rather than providing random guesses. To accommodate this preference, two types of accuracies were considered in this study, the first counts the

correct predictions only and the second extends the first one by also considering the instances classified to Ω. The two accuracies are expressed by the following formulas, respectively:

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions} \tag{4.1}$$

$$Extended\ accuracy = \frac{Number\ of\ correct\ predictions + Number\ of\ instances\ assigned\ to\ \Omega}{Total\ number\ of\ predictions} \tag{4.2}$$

4.4.2 Misclassification rate

The misclassification rate captures instances where the model inaccurately assigns input data to classes other than their true ones, thereby reflecting the overall performance in making erroneous predictions. Analogous to the accuracy metric, an extended version of the misclassification rate was made to take into account the instances assigned to Ω as correct classifications. The two rates are given by:

$$Misclassification\ rate = 1 - accuracy \tag{4.3}$$

and

$$Extended\ misclassification\ rate = 1 - extended\ accuracy \tag{4.4}$$

4.4.3 Empirical cumulative distribution function (ECDF)

The empirical cumulative distribution function (ECDF) is a valuable statistical tool for assessing uncertainty, and this research applied it to the entropy of predictions. Entropy is one of the methods for measuring uncertainty in a probability distribution. ECDF allows to visualize the cumulative probability of observing entropy values up to a given threshold. A steep rise in the ECDF curve indicates a higher concentration of low entropies, suggesting more confident predictions. Conversely, a gradual ascent in the ECDF curve highlights a broader distribution of entropy values, signifying increased uncertainty in the model predictions. The entropy of predictions for data instances is computed by:

$$H(P) = -\sum_{i} P(i)\ \log(P(i)) \tag{4.5}$$

where $H(P)$ denotes the entropy and P is the prediction. ECDF can be found by:

$$F(x) = \frac{1}{N}\sum_{i=1}^{N} I(\ X_i\ \leq x\ ) \tag{4.6}$$

where $F(x)$ represents the ECDF value at entropy x, X_i is the i^{th} entropy value, and $I(\ X_i\ \leq x)$ is an indicator function that equals 1 when $X_i\ \leq x$ is true and 0 otherwise.

4.5 Code Implementation

All implementation of models, experiments, and evaluations were in Python on Google Colab, and the selections of GPUs were based on the tasks, e.g., a faster GPU for model training. Tensorflow [87] was used to implement d-EDL and other models included for evaluation.

Chapter 5

Results and Discussion

The previous two chapters delved into the methodology and implementation of the proposed evidential classifier, sharing experiment setups and describing evaluation metrics. This chapter delves deeper into the comprehensive analysis of results obtained from assessing models in three distinct scenarios: noise, image rotation, and OOD data. The assessment compares five variants of the proposed d-EDL classifier, each characterized by a different number of data representatives (20, 40, 60, 80, and 100), against Standard CNN, EDL, and m-EDL. Refer to Section 4.2.2 for detailed descriptions of the architectures and implementation specifics of the latter three models. The EDL threshold was set at 0.5, classifying any data point with uncertainty exceeding this threshold into Ω. This choice is intuitive, representing a midpoint between [0,1], where the extremes correspond to complete certainty and total uncertainty.

5.1 Classification Accuracy

Before assessing the performance of the proposed classifier on unfamiliar data, it is important to evaluate its performance on familiar data as a conventional classifier. This step is essential because if the model fails to discern known data patterns, its performance on unfamiliar data may be unreliable. The accuracies on the MNIST test set for all classifiers, alongside the number of instances classified as Ω, are reported in Table 5.1. In this accuracy calculation, any data instance assigned to Ω was considered a misclassification, as the models here dealt with familiar data and were expected to distinguish between data classes effectively. In the table, the first-best performance is highlighted in black bold text, while the second-best performance is in blue bold text.

Standard CNN attains the highest accuracy, outperforming other models. Comparatively, d-EDLs and m-EDL exhibit similar performance levels, while EDL records the lowest accuracy. For d-EDL models, the accuracy improves as n increases from 20 to 60, accompanied by a substantial reduction in instances assigned to Ω. This trend suggests improved learning with an increasing number of representatives. However, for d-EDL with 80 and 100 representatives, both accuracy and the number of instances decrease, indicating potential misclassification. Overall, all d-EDL variants achieve an accuracy exceeding 98%, showcasing their excellence in classifying familiar data patterns.

Table 5.1: Accuracies (%) and numbers of instances classified into Ω from the MNIST test set by the models.

Model	**Accuracy (%)**	**Num. of instances Classified to Ω**
Standard CNN	**99.02**	–
EDL	97.81	279
m-EDL	98.90	61
d-EDL ($n = 20$)	98.15	338
d-EDL ($n = 40$)	98.55	131
d-EDL ($n = 60$)	**98.93**	66
d-EDL ($n = 80$)	98.90	**60**
d-EDL ($n = 100$)	98.76	**52**

5.2 Noisy Data

This section aims to evaluate the capacity of the models to estimate uncertainty rather than making random guesses when confronted with noise. Noise is one of the most common factors leading to misclassification, as its presence can alter image pixel values, deceiving models. The higher the noise level, the greater the expected distortion in the visual content of the image. As σ increases, models may struggle to distinguish signal from noise, resulting in decreased performance and increased uncertainty. To illustrate this effect, Fig. 5.1 shows distorted images of the digit three from the MNIST dataset with added Gaussian noise, featuring varying σ values within the range [0,1]. In the

interval [0,0.1], small amounts of white noise are introduced, and the digit 3 remains recognizable. As σ increases from 0.2, more white spots are added, intensifying the distortion until the complete distortion is reached at σ=1.

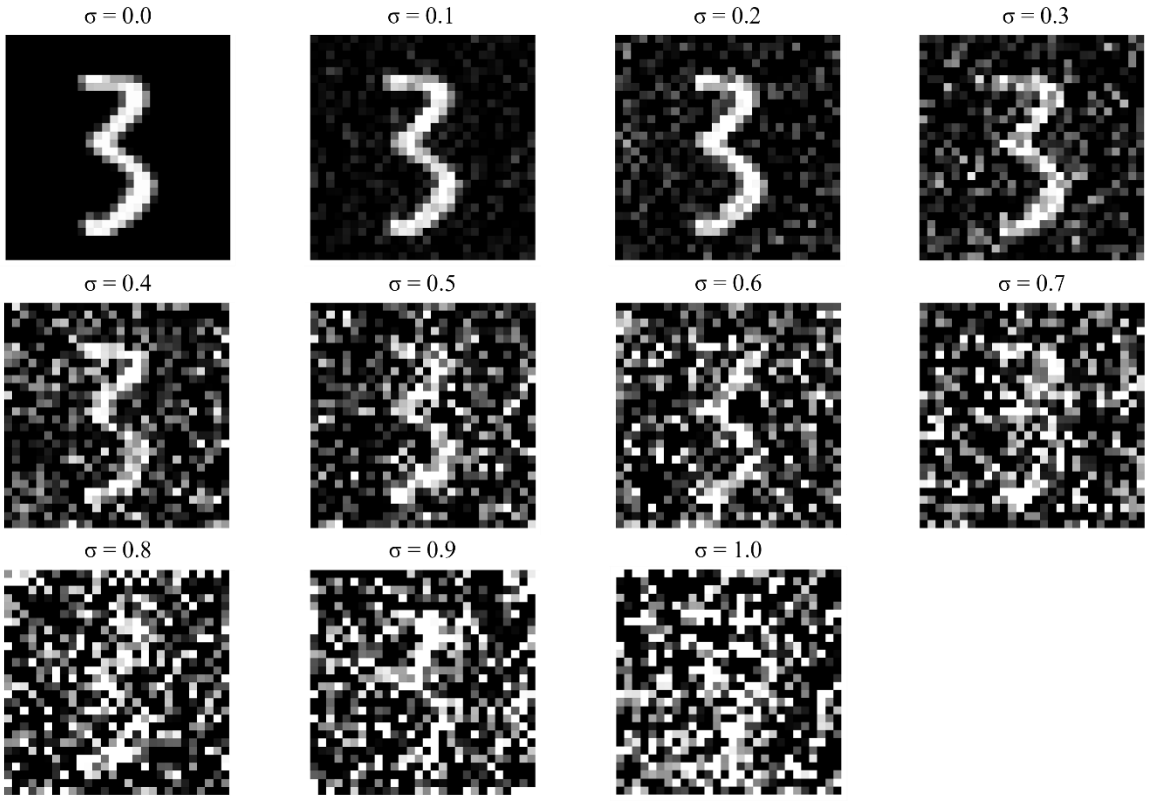
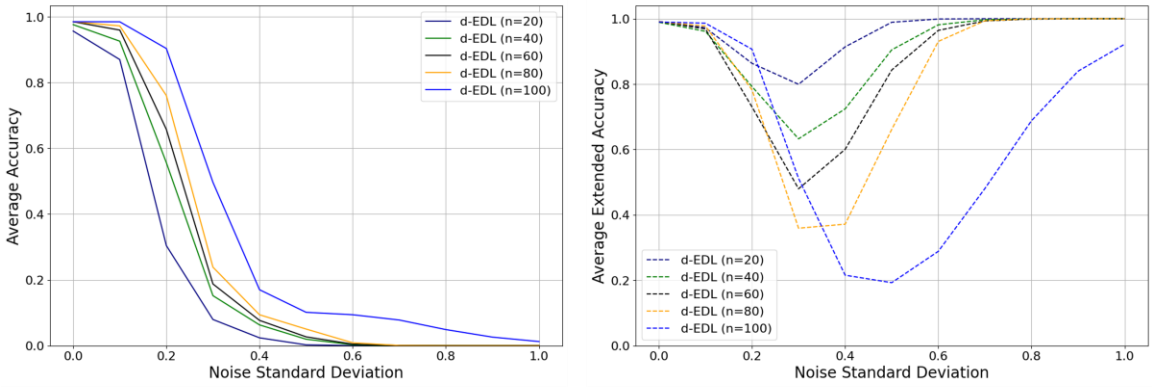


Figure 5.1: Distorted versions of an image of digit 3 from the MNIST dataset by Gaussian noise with different standard deviations.

The ability of d-EDLs to make precise predictions despite the presence of noise can be inferred from Fig. 5.2(a), which plots the average accuracy against the noise standard deviation within the range of [0,1]. As expected, a decline in accuracy is observed as σ increases. The accuracy curves begin to converge to 0 from $\sigma = 0.5$ onwards, indicating that the models struggle to discern distinct patterns, posing a challenge for correct

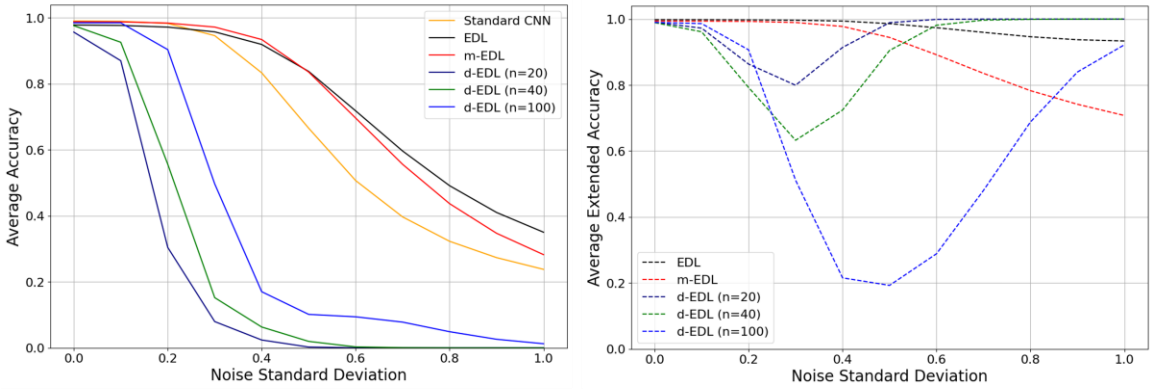
classification. The most decrease is noted for d-EDL with $n = 20$ for $\sigma \geq 0.2$. Meanwhile, d-EDL with 100 prototypes shows the slowest decay rate, declining after $\sigma = 0.2$ and achieving the highest accuracies for all σ values. The performances of the other variants fall between these two; as n decreases, the accuracy also decreases. In terms of making precise classification, it can be concluded that d-EDL with 20 representatives is the most sensitive to noise, while d-EDL with 100 representatives is the least sensitive.

As the noise level increases, the accuracy for the correct class decreases, raising the question of which class the incorrectly classified data points are assigned to. Therefore, Fig. 5.2(b) plots the average extended accuracy, counting data points assigned to Ω as correct classification against varying σ between [0,1]. Extended accuracy provides insights into the model effectiveness in assigning input images to their correct labels and to the uncertainty class. Based on the figure, as n decreases, the model exhibits higher uncertainty quantification ability, with d-EDLs having 20 and 100 representations being the best and worst quantifiers, respectively. d-EDLs exhibit the lowest accuracy values for σ ranging between [0.2,0.4], indicating that noise has succeeded in deceiving models, resulting in misclassification. The convergence to 0% accuracy for the σ region between [0.5,1] in Fig. 5.2(a) is mirrored by convergence to 100% accuracy in Fig. 5.2(b) for all d-EDLs except the one with 100 data representatives.



(a) (b)

Figure 5.2: Average accuracy versus noise standard deviation within the range [0,1] for d-EDL variants evaluated on noisy versions of the MNIST test set: (a) average accuracy, and (b) average extended accuracy.



(a) (b)

Figure 5.3: Average accuracy versus noise standard deviation within the range [0,1] for all models evaluated on noisy versions of the MNIST test set: (a) average accuracy, and (b) average extended accuracy.

From Fig. 5.2, it is evident that as n increases, the accuracy of correct classification improves, but it comes at the expense of the model ability to estimate uncertainty. Therefore, there is a trade-off between accurately classifying and effectively estimating

uncertainty. This behavior can be justified as follows: although it may seem intuitive that having more data representatives would result in better performance due to an increased number of evidence sources to rely on, this is not necessarily true in all cases. In this study, the trade-off could result from applying the second rule of combination, given in Eq. (3.9) and Eq. (3.10). In this rule, a combination of multiplication and addition operations is applied to combine belief masses for data classes, while only a multiplication operation is used to combine masses for class Ω. Multiplying two probabilities yields a smaller number than the number resulting from adding two probabilities. Hence, the higher the number of representatives in a model, the smaller the probability for the uncertainty class, which explains the observed behavior.

After a thorough evaluation of various versions of the proposed classifier, the next step is to assess its performance against other classifiers, namely Standard CNN, EDL, and m-EDL. Analogous to Fig. 5.2, Fig. 5.3 (a) and (b) present plots of average accuracy and average extended accuracy versus the standard deviation of noise, respectively. In this comparison, d-EDLs with $n = 20$ and 40 are considered, as they are identified as the best uncertainty estimators, along with the 100-representative-based d-EDL due as its high average accuracy. In Fig. 5.3(a), EDL and its modified version, m-EDL, achieve the highest accuracies overall, with EDL performing better from $\sigma = 0.5$ onwards, while d-EDLs exhibit the lowest accuracies.

In Fig. 5.3(b), the performances of all models are significantly improved across all σ values. EDL and m-EDL achieve accuracies of approximately 100% for $\sigma \leq 0.4$. Beyond $\sigma = 0.4$, their accuracies gradually decline to almost 94% and 70% for EDL and m-EDL, respectively. On the other hand, d-EDLs with $n = 20$ and 40 outperform the

other models at $\sigma \geq 0.6$, achieving accuracies of almost 100%. Hence, for low levels of noise ($\sigma = [0,0.4]$), EDL outperforms the others, and m-EDL comes in second. For high levels of noise ($\sigma = [0.6,1]$), d-EDLs with 20 and 40 representatives exhibit the best uncertainty estimation ability. EDL and d-EDL with $n = 20$ exhibit closely comparable performance at $\sigma = 0.5$, marking them the best performance at this level of noise. The notable performance difference between d-EDLs and the other models in Fig. 5.3(a) may be attributed to the metric employed for measuring the similarities between two feature vectors. The assumption is, the distance-based metric appears to have increased the model sensitivity to subtle changes introduced by the noise, nevertheless, a thorough investigation has to be conducted to find out the underlying cause. Most importantly, this heightened sensitivity has not resulted in misclassification, as evidenced by Fig. 5.3(b).

5.3 Image Rotation

Rotation mimics scenarios where images of an object are captured from different angles, leading to variations in its appearance in the images. Despite common 2D plane rotation techniques used in data augmentation, they fall short of fully representing the diverse angles present in a 3D space. Considering all possible variations in the appearance of a 3D object resulting from capturing images at different angles is impractical. Therefore, it is essential to equip models with the ability to estimate uncertainty when faced with unseen angles. In this evaluation, models trained on the MNIST training set without any data augmentation applied were assessed for their capacity to express uncertainty rather than making random classification. The models included in this assessment are d-EDLs

with n = 20, 40, and 100, as well as CNN, EDL, and m-EDL. The misclassification rate versus rotation angle between [0°,360°] is plotted in Fig. 5.4. In Fig. 5.4(a), only the true label is counted as a correct classification, while Fig. 5.4(b) considers both the true label and the uncertainty class to be correct answers.

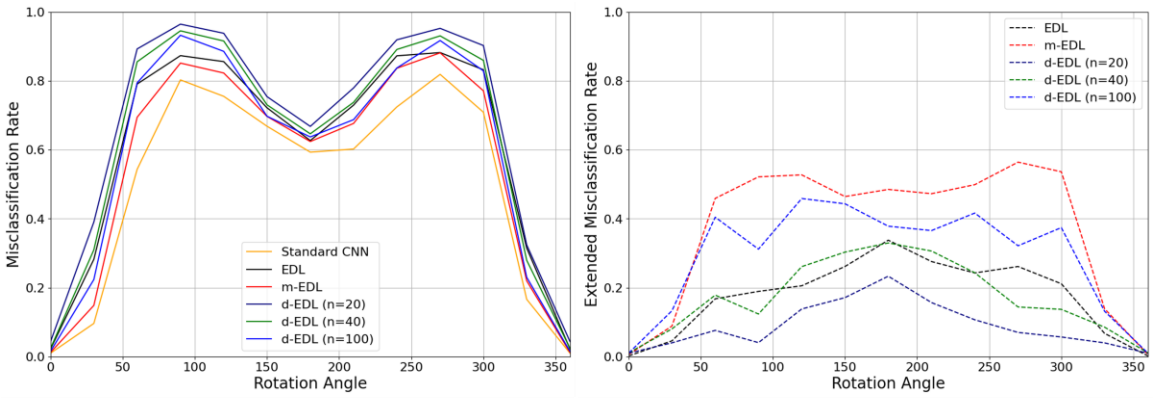


Figure 5.4: Misclassification rate versus various rotation angle ranging [0°,360°] obtained for all models evaluated on rotated images from the MNIST test set: (a) misclassification rate, and (b) extended misclassification rate.

In Fig. 5.4(a), the optimal performance is observed at $\theta \leq 30$ and $\theta \geq 330$ for all models, resulting in the lowest misclassification rates. Challenges arise in recognizing rotated digits for angles between [60°,120°] and [240°,300°], reflected in the highest misclassification rates. In these angle ranges, CNN exhibits the best performance, followed by m-EDL. EDL and d-EDL with $n = 100$ compete for the third position. As an uncertainty estimator, d-EDL with 20 representatives excels, outperforming the state-of-the-art classifier, EDL, which competes with d-EDL ($n = 40$) for the second position as shown in Fig. 5.4(b). The overall performance of d-EDLs with varying n is, as the

number of representatives decreases, the extended misclassification rate improves. This is an indication that the data points that were not assigned to their true labels, were successfully assigned to the uncertainty class.

5.4 OOD Data

This research aims to enable models to effectively express uncertainty when encountering unfamiliar data. Therefore, it is substantial to assess the model capability with OOD data. The ECDFs versus entropy of predictions is illustrated in Fig. 5.5 (a), (b), and (c) for the Fashion MNIST, EMNIST, and KMNIST datasets, respectively. The ECDFs were computed for d-EDL with $n = 20$, 40, and 100, as well as Standard CNN, EDL, and m-EDL. The maximum entropy is achieved when the predicted probabilities for all classes are equal. This corresponds to a state of maximum uncertainty, where the model cannot distinguish between the classes and assigns an equal probability to each. On the other hand, the minimum entropy occurs when the model is perfectly confident in its predictions, assigning a probability of one to the correct class and zero to all other classes. When dealing with OOD data, higher entropies are desired, and the closer the curve is to the bottom right corner, the better.

Both m-EDL and d-EDL incorporate the uncertainty class into the probability distribution, assigning a probability to Ω, satisfying the constraint $(\sum_{j=1}^{M} P(\omega_j)) + P(\Omega) = 1$. This is in contrast to EDL, where uncertainty is provided separately with a value ranging between [0,1]. Assigning a high probability to the uncertainty class results in a low entropy value for m-EDL and d-EDL, implying low uncertainty, which is untrue.

The probability assigned to Ω reflects how uncertain the classifier is about the correct classification, and the correct label could be any class within or outside the true label set. To align the results with Standard CNN and EDL, the pignistic transformation expressed in Eq. (2.10) was applied to distribute the probability of Ω equally among all classes in the dataset.

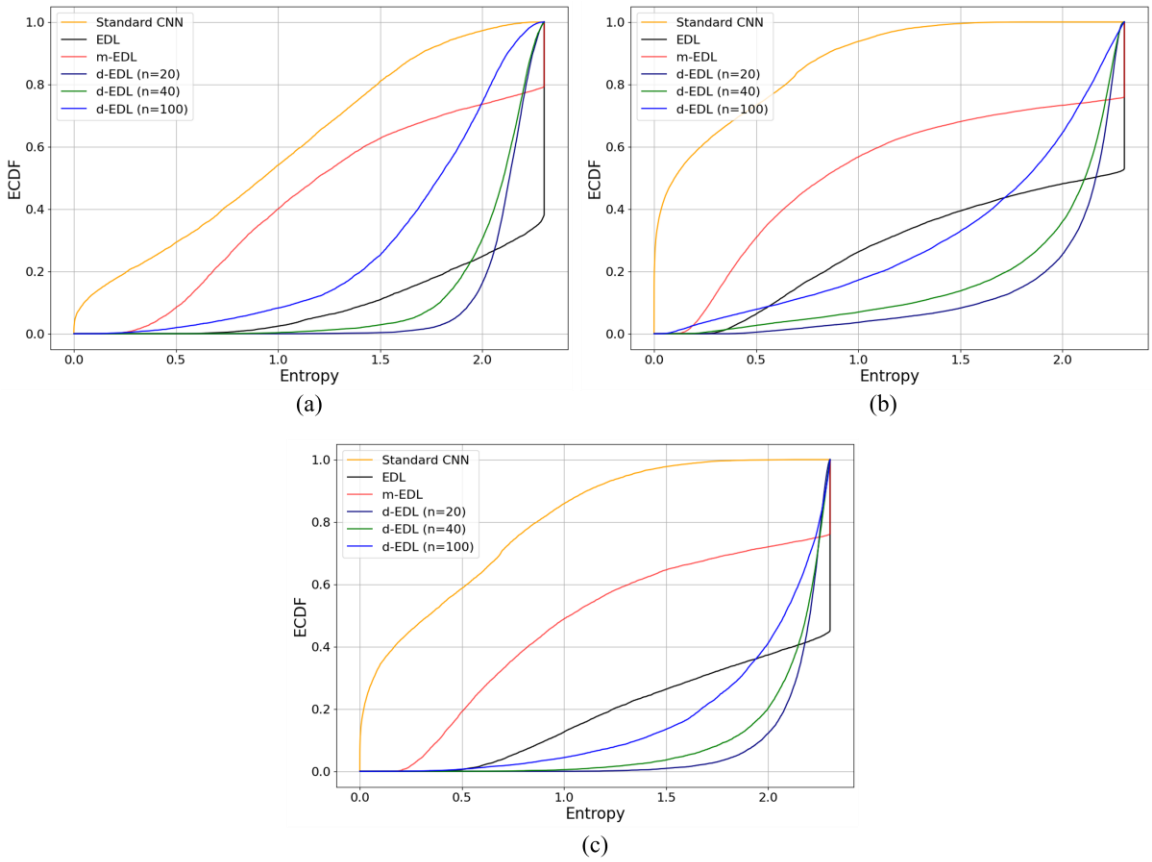


Figure 5.5: ECDF versus entropy of predictions on the test sets from three datasets: (a) Fashion MNIST, (b) EMNIST, and (c) KMINST.

The ECDF-entropy plots in Fig. 5.5 reveal that d-EDLs with 20 and 40 representatives and EDL generate the highest entropies. The two d-EDLs exhibit highly

comparable performance to the state-of-the-art EDL for the Fashion MNIST dataset (Fig. 5.5(a)). Interestingly, they outperform EDL for the EMNIST and KMINST datasets, as observed in Fig. 5.5(b) and Fig. 5.5(c), respectively. Even d-EDL with a hundred data representatives demonstrates an excellent ability to generate high entropies for all datasets, competing with EDL for the EMNIST and KMNIST datasets.

Table 5.2 provides the percentages of instances assigned to the uncertainty class by each model for the three datasets, relative to the total number of instances in the dataset. A higher percentage is desirable, with the top-performing model highlighted in black bold text, the second-best in blue bold text, and the third in green bold text. Notably, d-EDL ($n = 20$) achieves the highest percentages across all datasets, and d-EDL ($n = 40$) secures the second position for EMNIST and KMNIST datasets. The percentages attained by d-EDL with 100 representatives for EMNIST and KMNIST are comparable to those achieved by the state-of-the-art models, especially for KMNIST. These findings along with the achieved high entropies by d-EDLs prove the excellence of the proposed architecture to recognize out-of-distribution data.

To examine the hypothesis that models face challenges recognizing OOD data resembling the training data, Table 5.3 presents the mean and standard deviation of the Structural Similarity Index (SSIM) [88] for pairs of distinct image datasets: MNIST, Fashion MNIST, EMNIST, and KMNIST. The similarity measurements involved the comparisons of 500 images from each class, amounting to 5,000 images per dataset except for EMNIST. There are 47 classes in the EMNIST dataset, equal numbers of images were taken from these classes, making in total nearly 5,000 images for the entire dataset. Each image in a dataset was compared to 5,000 images from the other datasets.

The mean SSIM, along with its standard deviation (in parentheses), serves as a measure of similarity, where a higher mean SSIM indicates relatively greater similarity, and a larger standard deviation reflects increased variability. Based on SSIM means, EMNIST exhibits the highest similarity to MNIST. Fashion MNIST and KMNIST have mean SSIM scores that are nearly equivalent and rank them second and third, respectively.

Table 5.2: Percentages (%) of data instances classified into Ω to the total number of instances for Fashion MNIST, EMNIST, and KMNIST.

Model	Fashion MNIST	EMNIST	KMNIST
EDL	**84.25**	**56.97**	**69.43**
m-EDL	26.24	26.68	27.90
d-EDL ($n = 20$)	**91.15**	**79.48**	**92.29**
d-EDL ($n = 40$)	**83.65**	**71.42**	**84.91**
d-EDL ($n = 100$)	28.37	45.99	67.54

Multidimensional Scaling (MDS) was employed to visualize the proximity of datasets to each other (see Fig. 5.6), where the distance between two datasets is represented as the inverse of SSIM. Consistent with the findings in Table 5.3, EMNIST emerges as the closest to MNIST, followed by Fashion MNIST and KMNIST. Notably, models demonstrate the lowest performance in expressing uncertainty for the EMNIST dataset compared to the other datasets, supporting the initial hypothesis.

Table 5.3: The mean (standard deviation) of SSIM values for the four datasets: MNIST, Fashion MNIST, EMNIST, and KMNIST.

	MNIST	**Fashion MNIST**	**EMNIST**	**KMNIST**
MNIST	–	0.1066 (0.0945)	0.1332 (0.1109)	0.1006 (0.1042)
Fashion MNIST	–	–	0.0915 (0.0831)	0.0614 (0.0808)
EMNIST	–	–	–	0.0736 (0.0926)
KMNIST	–	–	–	–

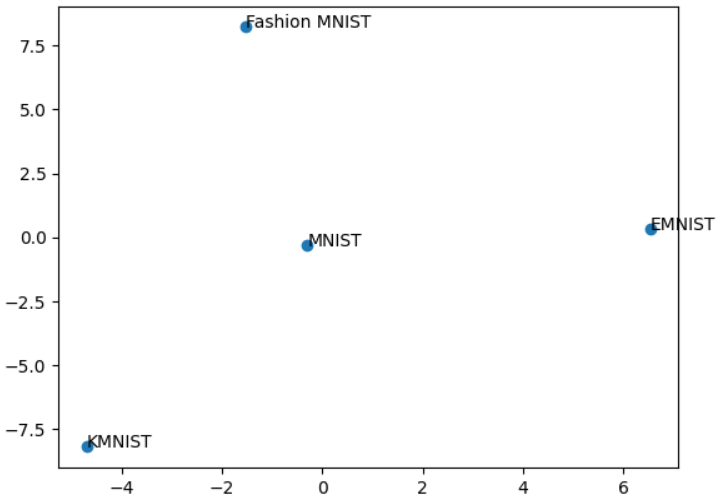


Figure 5.6: Locations of MNIST, Fashion MNIST, EMNIST, and KMNIST estimated via MDS, and the distances between points represent the inverse of the similarity means. The axes are dimensionless.

Chapter 6

Conclusions and Future Work

This chapter summarizes the research and highlights the significant outcomes and research impacts. Furthermore, it shares the limitations of the studies and provides future directions.

6.1 Research Impacts

This study introduces an evidential deep learning classifier, d-EDL, that leverages the powerful capabilities of CNNs for feature extraction. The d-EDL incorporates specially designed decision-making layers, replacing the last two layers typically found in CNNs—specifically, the fully connected and softmax layers. The designed layers compute BPAs by measuring similarities between input data and formulated data representatives, then merging these BPAs to form the network final decision. The output is an extended probability distribution, incorporating an uncertainty class whose assigned probability reflects the model uncertainty about the classification of the input data pattern. Moreover,

the research provides an end-to-end training method for the proposed network to optimize all network parameters jointly. In addition, the thesis proposes experiments to assess the model capabilities when confronted with common uncertainty-inducing factors (noise, image rotation, and out-of-distribution data), mimicking real-world scenarios. The findings from these various scenarios demonstrate the great capability of d-EDLs to estimate uncertainty when faced, especially d-EDLs with lower numbers of data representatives.

This research significantly impacts the field of DNNs by introducing a novel network architecture with an added ability to quantify uncertainty effectively, enabling safer decisions compared to typical DNNs. This ability enhances the reliability of DNNs, making them more suitable for safety-critical applications. Furthermore, this research stands out as one of the few studies focusing on enabling uncertainty quantification within the network architectures, allowing for automatic estimation. Therefore, the designed classifier architecture and training method can serve as a guide for future research, demonstrating how to incorporate uncertainty quantification methods within network architectures. In addition, the guide shows how to optimize the parameters contributing to quantification during training. Most research focuses on evaluating models, mainly on OOD data; this research extends the assessment to include noise and image rotation—other common factors that can potentially lead to misclassification. Consequently, the research provides additional experiments mimicking real-life scenarios, offering further insights into the network performance.

6.2 Limitations and Future Directions

The outcomes of the conducted experiments confirm that d-EDL successfully attains its primary objective— effective uncertainty estimation when uncertainty is encountered. However, these findings come with certain limitations, highlighting the need for future refinement. Consequently, this section identifies limitations and provides directions for future enhancements, aiming to improve the effectiveness and applicability of the proposed approach. The limitations and suggested directions for future improvements are outlined below:

- A trade-off is noted between classifying to the true labels and uncertainty quantification when handling noisy images. As the number of data representatives decreases, d-EDL becomes better at quantifying uncertainty but less accurate at classifying data into the correct classes. The trade-off is a result of applying the second combination rule to merge BPAs. Thus, future research should explore methods to overcome this limitation, applying approaches to combine BPAs in a more meaningful and intuitive way.
- Compared to other models in related work, d-EDL appears to be more sensitive to changes introduced by noise, resulting in lower accuracies for classifying the noisy images into their true labels. A thorough investigation must be carried out to determine the cause of this sensitivity to resolve it and enhance the robustness of the proposed architecture.

- The designed decision-making layers were integrated into a single architecture of standard baseline CNNs, and the decisions heavily rely on measuring similarities between extracted feature vectors by the CNNs. Subsequent research could delve into examining how diverse CNN architectures, such as those varying in depth and width and producing different feature vectors, impact the decision-making process.
- The proposed method of converting features into an extended probability distribution was only incorporated into one type of CNNs and tested with one dataset. Hence, future studies must investigate the scalability of the proposed approach to other CNNs and datasets with varying sizes.
- The study did not apply optimization techniques to find the optimal number of data representatives for each class; rather, it unified the numbers for all classes. Future research could explore methods to dynamically determine the best number of representatives for each class. This can be extremely advantageous when dealing with imbalanced data.

Bibliography

[1] S. Dua *et al.*, “Developing a Speech Recognition System for Recognizing Tonal Speech Signals Using a Convolutional Neural Network,” *Appl. Sci.*, vol. 12, no. 12, p. 6223, Jun. 2022, doi: 10.3390/app12126223.

[2] M. Coskun, A. Ucar, O. Yildirim, and Y. Demir, “Face recognition based on convolutional neural network,” in *2017 International Conference on Modern Electrical and Energy Systems (MEES)*, Kremenchuk: IEEE, Nov. 2017, pp. 376–379. doi: 10.1109/MEES.2017.8248937.

[3] D. J. Hemanth, J. Anitha, A. Naaji, O. Geman, D. E. Popescu, and L. Hoang Son, “A Modified Deep Convolutional Neural Network for Abnormal Brain Image Classification,” *IEEE Access*, vol. 7, pp. 4275–4283, 2019, doi: 10.1109/ACCESS.2018.2885639.

[4] Q. Li, W. Cai, X. Wang, Y. Zhou, D. D. Feng, and M. Chen, “Medical image classification with convolutional neural network,” in *2014 13th International Conference on Control Automation Robotics & Vision (ICARCV)*, Singapore: IEEE, Dec. 2014, pp. 844–848. doi: 10.1109/ICARCV.2014.7064414.

[5] S. OwaisAli Chishti, S. Riaz, M. BilalZaib, and M. Nauman, “Self-Driving Cars Using CNN and Q-Learning,” in *2018 IEEE 21st International Multi-Topic*

Conference (INMIC), Karachi: IEEE, Nov. 2018, pp. 1–7. doi: 10.1109/INMIC.2018.8595684.

[6] G. Latif, S. E. Abdelhamid, R. E. Mallouhy, J. Alghazo, and Z. A. Kazimi, "Deep Learning Utilization in Agriculture: Detection of Rice Plant Diseases Using an Improved CNN Model," *Plants*, vol. 11, no. 17, p. 2230, Aug. 2022, doi: 10.3390/plants11172230.

[7] A. Mandelbaum and D. Weinshall, "Distance-based Confidence Score for Neural Network Classifiers," 2017, doi: 10.48550/ARXIV.1709.09844.

[8] D. P. Bertsekas and J. N. Tsitsiklis, *Introduction to probability*, 2nd ed. in Optimization and computation series. Belmont: Athena scientific, 2008.

[9] S. M. Ross, *Introduction to probability and statistics for engineers and scientists*, 4th ed. Amsterdam: Academic Press/Elsevier, 2009.

[10] M. Holický, *Introduction to Probability and Statistics for Engineers*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. doi: 10.1007/978-3-642-38300-7.

[11] M. Hafeez, "Application of Dempster Shafer Theory to Assess the Status of Sealed Fire in a Cole Mine," Blekinge Institute of Technology, Sweden, 2011. [Online]. Available: http://www.diva-portal.org/smash/record.jsf?pid=diva2%3A832697&dswid=-664

[12] A. P. Dempster, "Upper and Lower Probabilities Induced by a Multivalued Mapping," *Ann. Math. Stat.*, vol. 38, no. 2, pp. 325–339, Apr. 1967, doi: 10.1214/aoms/1177698950.

[13] G. Shafer, *A Mathematical Theory of Evidence*. Princeton University Press, 2020. doi: 10.2307/j.ctv10vm1qb.

[14] P. Smets and R. Kennes, “The transferable belief model,” *Artif. Intell.*, vol. 66, no. 2, pp. 191–234, Apr. 1994, doi: 10.1016/0004-3702(94)90026-4.

[15] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998, doi: 10.1109/5.726791.

[16] L. Alzubaidi *et al.*, “Review of deep learning: concepts, CNN architectures, challenges, applications, future directions,” *J. Big Data*, vol. 8, no. 1, p. 53, Mar. 2021, doi: 10.1186/s40537-021-00444-8.

[17] D. Bhatt *et al.*, “CNN Variants for Computer Vision: History, Architecture, Application, Challenges and Future Scope,” *Electronics*, vol. 10, no. 20, p. 2470, Oct. 2021, doi: 10.3390/electronics10202470.

[18] D. E. Rumelhart, G. E. Hintont, and R. J. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[19] J. Mena, O. Pujol, and J. Vitrià, “A Survey on Uncertainty Estimation in Deep Learning Classification Systems from a Bayesian Perspective,” *ACM Comput. Surv.*, vol. 54, no. 9, pp. 1–35, Dec. 2022, doi: 10.1145/3477140.

[20] Y. Gal and Z. Ghahramani, “Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning,” 2015, doi: 10.48550/ARXIV.1506.02142.

[21] R. Senge *et al.*, “Reliable classification: Learning classifiers that distinguish aleatoric and epistemic uncertainty,” *Inf. Sci.*, vol. 255, pp. 16–29, Jan. 2014, doi: 10.1016/j.ins.2013.07.030.

[22] J. Mena, O. Pujol, and J. Vitria, “Uncertainty-Based Rejection Wrappers for Black-Box Classifiers,” *IEEE Access*, vol. 8, pp. 101721–101746, 2020, doi: 10.1109/ACCESS.2020.2996495.

[23] J. Yu, D. Wang, and M. Zheng, “Uncertainty quantification: Can we trust artificial intelligence in drug discovery?,” *iScience*, vol. 25, no. 8, p. 104814, Aug. 2022, doi: 10.1016/j.isci.2022.104814.

[24] R. M. Neal, *Bayesian learning for neural networks*. Springer New York, 2012.

[25] S. Ryu, Y. Kwon, and W. Y. Kim, “A Bayesian graph convolutional network for reliable prediction of molecular properties with uncertainty quantification,” *Chem. Sci.*, vol. 10, no. 36, pp. 8438–8446, 2019, doi: 10.1039/C9SC01992H.

[26] E. Goan and C. Fookes, “Bayesian Neural Networks: An Introduction and Survey,” in *Case Studies in Applied Bayesian Data Science*, vol. 2259, K. L. Mengersen, P. Pudlo, and C. P. Robert, Eds., in Lecture Notes in Mathematics, vol. 2259. , Cham: Springer International Publishing, 2020, pp. 45–87. doi: 10.1007/978-3-030-42553-1_3.

[27] D. J. C. MacKay, “A Practical Bayesian Framework for Backpropagation Networks,” *Neural Comput.*, vol. 4, no. 3, pp. 448–472, May 1992, doi: 10.1162/neco.1992.4.3.448.

[28] M. E. Khan, A. Immer, E. Abedi, and M. Korzepa, “Approximate Inference Turns Deep Networks into Gaussian Processes,” 2019, doi: 10.48550/ARXIV.1906.01930.

[29] H. Ritter, A. Botev, and D. Barber, “A Scalable Laplace Approximation for Neural Networks,” in *ICL 2018-conference track proceedings: 6th International Conference*

on Learning Representations, 2018. [Online]. Available: https://openreview.net/forum?id=Skdvd2xAZ

[30] J. M. Hernández-Lobato and R. P. Adams, “Probabilistic Backpropagation for Scalable Learning of Bayesian Neural Networks,” 2015, doi: 10.48550/ARXIV.1502.05336.

[31] J. Martens and R. Grosse, “Optimizing Neural Networks with Kronecker-factored Approximate Curvature,” 2015, doi: 10.48550/ARXIV.1503.05671.

[32] A. Y. K. Foong, Y. Li, J. M. Hernández-Lobato, and R. E. Turner, “‘In-Between’ Uncertainty in Bayesian Neural Networks.” arXiv, Jun. 27, 2019. Accessed: Jan. 11, 2024. [Online]. Available: http://arxiv.org/abs/1906.11537

[33] A. Immer, M. Korzepa, and M. Bauer, “Improving predictions of Bayesian neural nets via local linearization,” arXiv, 2020. doi: 10.48550/ARXIV.2008.08400.

[34] Z. Deng, F. Zhou, and J. Zhu, “Accelerated Linearized Laplace Approximation for Bayesian Deep Learning.” arXiv, Oct. 23, 2022. Accessed: Jan. 29, 2024. [Online]. Available: http://arxiv.org/abs/2210.12642

[35] S. S. Qian, C. A. Stow, and M. E. Borsuk, “On Monte Carlo methods for Bayesian inference,” *Ecol. Model.*, vol. 159, no. 2–3, pp. 269–277, Jan. 2003, doi: 10.1016/S0304-3800(02)00299-5.

[36] K. Gallagher, K. Charvin, S. Nielsen, M. Sambridge, and J. Stephenson, “Markov chain Monte Carlo (MCMC) sampling methods to determine optimal models, model resolution and model choice for Earth Science problems,” *Mar. Pet. Geol.*, vol. 26, no. 4, pp. 525–535, Apr. 2009, doi: 10.1016/j.marpetgeo.2009.01.003.

[37] M. Welling and Y. W. Teh, “Bayesian learning via stochastic gradient langevin dynamics,” presented at the International Conference on Machine Learning, 2011, pp. 681–688.

[38] F. Liang, J. Kim, and Q. Song, “A Bootstrap Metropolis–Hastings Algorithm for Bayesian Analysis of Big Data,” *Technometrics*, vol. 58, no. 3, pp. 304–318, Jul. 2016, doi: 10.1080/00401706.2016.1142905.

[39] S. Chib and E. Greenberg, “Understanding the Metropolis-Hastings Algorithm,” *Am. Stat.*, vol. 49, no. 4, p. 327, Nov. 1995, doi: 10.2307/2684568.

[40] R. Zhang, A. F. Cooper, and C. De Sa, “Asymptotically Optimal Exact Minibatch Metropolis-Hastings.” arXiv, Oct. 22, 2020. Accessed: Jan. 25, 2024. [Online]. Available: http://arxiv.org/abs/2006.11677

[41] R. Cornish, P. Vanetti, A. Bouchard-Côté, G. Deligiannidis, and A. Doucet, “Scalable Metropolis-Hastings for Exact Bayesian Inference with Large Datasets.” arXiv, Jun. 10, 2019. Accessed: Jan. 25, 2024. [Online]. Available: http://arxiv.org/abs/1901.09881

[42] W. Zhang and R. Zhang, “DP-Fast MH: Private, Fast, and Accurate Metropolis-Hastings for Large-Scale Bayesian Inference.” arXiv, Oct. 12, 2023. Accessed: Jan. 25, 2024. [Online]. Available: http://arxiv.org/abs/2303.06171

[43] H. Abdulqadir Khidir, İ. Etikan, D. Hussein Kadir, N. H. Mahmood, and R. Sabetvand, “Bayesian machine learning analysis with Markov Chain Monte Carlo techniques for assessing characteristics and risk factors of Covid-19 in Erbil City-Iraq 2020–2021,” *Alex. Eng. J.*, vol. 78, pp. 162–174, Sep. 2023, doi: 10.1016/j.aej.2023.07.052.

[44] M. Betancourt, “A Conceptual Introduction to Hamiltonian Monte Carlo.” arXiv, Jul. 15, 2018. Accessed: Jan. 28, 2024. [Online]. Available: http://arxiv.org/abs/1701.02434

[45] L. Yang, X. Meng, and G. E. Karniadakis, “B-PINNs: Bayesian physics-informed neural networks for forward and inverse PDE problems with noisy data,” *J. Comput. Phys.*, vol. 425, p. 109913, Jan. 2021, doi: 10.1016/j.jcp.2020.109913.

[46] A. Gelman, W. R. Gilks, and G. O. Roberts, “Weak convergence and optimal scaling of random walk Metropolis algorithms,” *Ann. Appl. Probab.*, vol. 7, no. 1, Feb. 1997, doi: 10.1214/aoap/1034625254.

[47] A. Olivier, M. D. Shields, and L. Graham-Brady, “Bayesian neural networks for uncertainty quantification in data-driven materials modeling,” *Comput. Methods Appl. Mech. Eng.*, vol. 386, p. 114079, Dec. 2021, doi: 10.1016/j.cma.2021.114079.

[48] T. Papamarkou, J. Hinkle, M. T. Young, and D. Womble, “Challenges in Markov chain Monte Carlo for Bayesian neural networks,” 2019, doi: 10.48550/ARXIV.1910.06539.

[49] J. Yao, W. Pan, S. Ghosh, and F. Doshi-Velez, “Quality of Uncertainty Quantification for Bayesian Neural Network Inference,” 2019, doi: 10.48550/ARXIV.1906.09686.

[50] T. Chen, E. B. Fox, and C. Guestrin, “Stochastic Gradient Hamiltonian Monte Carlo,” 2014, doi: 10.48550/ARXIV.1402.4102.

[51] C. Li, C. Chen, D. Carlson, and L. Carin, “Preconditioned Stochastic Gradient Langevin Dynamics for Deep Neural Networks,” 2015, doi: 10.48550/ARXIV.1512.07666.

[52] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, “On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima.” arXiv, Feb. 09, 2017. Accessed: Feb. 12, 2024. [Online]. Available: http://arxiv.org/abs/1609.04836

[53] Y. W. Teh, A. Thiéry, and S. Vollmer, “Consistency and fluctuations for stochastic gradient Langevin dynamics,” 2014, doi: 10.48550/ARXIV.1409.0578.

[54] C. Chen, N. Ding, and L. Carin, “On the Convergence of Stochastic Gradient MCMC Algorithms with High-Order Integrators,” 2016, doi: 10.48550/ARXIV.1610.06665.

[55] R. Zhang, C. Li, J. Zhang, C. Chen, and A. G. Wilson, “Cyclical Stochastic Gradient MCMC for Bayesian Deep Learning.” arXiv, May 11, 2020. Accessed: Feb. 12, 2024. [Online]. Available: http://arxiv.org/abs/1902.03932

[56] A. Ganguly and S. W. F. Earp, “An Introduction to Variational Inference.” arXiv, Nov. 21, 2021. Accessed: Jan. 28, 2024. [Online]. Available: http://arxiv.org/abs/2108.13083

[57] K. P. Murphy, *Machine learning: a probabilistic perspective*. in Adaptive computation and machine learning series. Cambridge, MA: MIT Press, 2012.

[58] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, “Variational Inference: A Review for Statisticians,” 2016, doi: 10.48550/ARXIV.1601.00670.

[59] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, “Weight Uncertainty in Neural Networks,” 2015, doi: 10.48550/ARXIV.1505.05424.

[60] P. Myshkov and S. Julier, “Posterior distribution analysis for Bayesian inference in neural networks,” *Workshop on Bayesian Deep Learning*, 2016, [Online]. Available: http://bayesiandeeplearning.org/2016/papers/BDL_34.pdf

[61] C. Louizos and M. Welling, “Structured and Efficient Variational Deep Learning with Matrix Gaussian Posteriors,” 2016, doi: 10.48550/ARXIV.1603.04733.

[62] A. K. Gupta and D. K. Nagar, *Matrix Variate Distributions*, 1st ed. Chapman and Hall/CRC, 2018. doi: 10.1201/9780203749289.

[63] C. Louizos and M. Welling, “Multiplicative Normalizing Flows for Variational Bayesian Neural Networks,” 2017, doi: 10.48550/ARXIV.1703.01961.

[64] N. Pawlowski, A. Brock, M. C. H. Lee, M. Rajchl, and B. Glocker, “Implicit Weight Uncertainty in Neural Networks,” 2017, doi: 10.48550/ARXIV.1711.01297.

[65] D. Ha, A. Dai, and Q. V. Le, “HyperNetworks.” arXiv, Dec. 01, 2016. Accessed: Jan. 29, 2024. [Online]. Available: http://arxiv.org/abs/1609.09106

[66] B. Lakshminarayanan, A. Pritzel, and C. Blundell, “Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles,” 2016, doi: 10.48550/ARXIV.1612.01474.

[67] T. Pearce, F. Leibfried, A. Brintrup, M. Zaki, and A. Neely, “Uncertainty in Neural Networks: Approximately Bayesian Ensembling,” 2018, doi: 10.48550/ARXIV.1810.05546.

[68] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy, “Deep Exploration via Bootstrapped DQN,” 2016, doi: 10.48550/ARXIV.1602.04621.

[69] Y. Ovadia *et al.*, “Can You Trust Your Model’s Uncertainty? Evaluating Predictive Uncertainty Under Dataset Shift,” 2019, doi: 10.48550/ARXIV.1906.02530.

[70] Y. Gal and Z. Ghahramani, “Bayesian Convolutional Neural Networks with Bernoulli Approximate Variational Inference,” 2015, doi: 10.48550/ARXIV.1506.02158.

[71] R. Seoh, “Qualitative Analysis of Monte Carlo Dropout,” 2020, doi: 10.48550/ARXIV.2007.01720.

[72] Y. Gal, J. Hron, and A. Kendall, “Concrete Dropout.” arXiv, May 22, 2017. Accessed: Oct. 31, 2023. [Online]. Available: http://arxiv.org/abs/1705.07832

[73] D. P. Kingma, T. Salimans, and M. Welling, “Variational Dropout and the Local Reparameterization Trick,” 2015, doi: 10.48550/ARXIV.1506.02557.

[74] S. Boluki, R. Ardywibowo, S. Z. Dadaneh, M. Zhou, and X. Qian, “Learnable Bernoulli Dropout for Bayesian Deep Learning.” arXiv, Feb. 12, 2020. Accessed: Oct. 31, 2023. [Online]. Available: http://arxiv.org/abs/2002.05155

[75] C. Theobald, F. Pennerath, B. Conan-Guez, M. Couceiro, and A. Napoli, “A Bayesian Neural Network based on Dropout Regulation.” arXiv, Feb. 03, 2021. Accessed: Oct. 31, 2023. [Online]. Available: http://arxiv.org/abs/2102.01968

[76] T. Denoeux, “A neural network classifier based on Dempster-Shafer theory,” *IEEE Trans. Syst. Man Cybern. - Part Syst. Hum.*, vol. 30, no. 2, pp. 131–150, Mar. 2000, doi: 10.1109/3468.833094.

[77] Z. Tong, P. Xu, and T. Denœux, “An evidential classifier based on Dempster-Shafer theory and deep learning,” *Neurocomputing*, vol. 450, pp. 275–293, Aug. 2021, doi: 10.1016/j.neucom.2021.03.066.

[78] M. Sensoy, L. Kaplan, and M. Kandemir, “Evidential Deep Learning to Quantify Classification Uncertainty,” 2018, doi: 10.48550/ARXIV.1806.01768.

[79] A. Nagahama, “Learning and predicting the unknown class using evidential deep learning,” *Sci. Rep.*, vol. 13, no. 1, p. 14904, Sep. 2023, doi: 10.1038/s41598-023-40649-w.

[80] Z. Zhang, J. Zhang, and H. Xue, “Improved K-Means Clustering Algorithm,” in *2008 Congress on Image and Signal Processing*, Sanya, China: IEEE, 2008, pp. 169–172. doi: 10.1109/CISP.2008.350.

[81] E. Umargono, J. E. Suseno, and S. K. Vincensius Gunawan, “K-Means Clustering Optimization Using the Elbow Method and Early Centroid Determination Based on Mean and Median Formula:,” in *Proceedings of the 2nd International Seminar on Science and Technology (ISSTEC 2019)*, Yogyakarta, Indonesia: Atlantis Press, 2020. doi: 10.2991/assehr.k.201010.019.

[82] Li Deng, “The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web],” *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 141–142, Nov. 2012, doi: 10.1109/MSP.2012.2211477.

[83] H. Xiao, K. Rasul, and R. Vollgraf, “Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms.” arXiv, Sep. 15, 2017. Accessed: Nov. 20, 2023. [Online]. Available: http://arxiv.org/abs/1708.07747

[84] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, “EMNIST: an extension of MNIST to handwritten letters.” arXiv, Mar. 01, 2017. Accessed: Nov. 20, 2023. [Online]. Available: http://arxiv.org/abs/1702.05373

[85] T. Clanuwat, M. Bober-Irizar, A. Kitamoto, A. Lamb, K. Yamamoto, and D. Ha, “Deep Learning for Classical Japanese Literature.” 2018. doi: 10.20676/00000341.

[86] “m-EDL.” Accessed: Oct. 11, 2023. [Online]. Available: https://github.com/naga0862/m-EDL

[87] M. Abadi *et al.*, “TensorFlow: A system for large-scale machine learning,” 2016, doi: 10.48550/ARXIV.1605.08695.

[88] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image Quality Assessment: From Error Visibility to Structural Similarity,” *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004, doi: 10.1109/TIP.2003.819861.